

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

HULU, LLC and
NETFLIX, INC.,
Petitioners,

v.

UNILOC 2017, LLC,
Patent Owner.

Case No. IPR2020-00041
Patent No. 8,407,609

**DECLARATION OF MICHAEL FRANZ IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW OF U.S. PATENT NO. 8,407,609**

TABLE OF CONTENTS

| | | |
|------|--|----|
| I. | INTRODUCTION | 1 |
| A. | Qualifications | 2 |
| 1. | Education | 2 |
| 2. | Work Experience..... | 2 |
| 3. | Publications..... | 5 |
| 4. | Curriculum Vitae..... | 6 |
| B. | Materials Reviewed..... | 6 |
| C. | Level of Ordinary Skill in the Art..... | 7 |
| D. | Summary of Opinions | 8 |
| II. | OVERVIEW OF THE TECHNOLOGY..... | 9 |
| A. | Priority Date of the Claims..... | 9 |
| B. | Overview of Relevant Technology When the '609 Patent Was Filed..... | 10 |
| 1. | Usage Tracking | 10 |
| 2. | Streaming Media..... | 18 |
| C. | The '609 Patent | 22 |
| D. | The Challenged Claims | 26 |
| E. | Claim Construction | 27 |
| 1. | “Computer System” | 28 |
| 2. | “Streamed” | 29 |
| III. | UNPATENTABILITY OF THE '609 PATENT CLAIMS | 30 |
| A. | Standards for Invalidity | 30 |
| 1. | Obviousness | 30 |
| B. | Ground I: Claims 1-3 were obvious in view of Davis and Choi..... | 32 |
| 1. | The Davis-Choi Combination..... | 32 |
| 2. | Claim 1 | 47 |

| | | |
|-----|---|-----|
| 3. | Claim 2: “The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.” | 74 |
| 4. | Claim 3: “The method of claim 2, wherein the received data is indicative of a temporal cycle passing.” | 78 |
| C. | Ground II: Claims 1-3 were obvious in view of Siler and Davis..... | 79 |
| 1. | The Siler-Davis Combination | 79 |
| 2. | Claim 1 | 92 |
| 3. | Claim 2: “The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.” | 103 |
| 4. | Claim 3: “The method of claim 2, wherein the received data is indicative of a temporal cycle passing.” | 107 |
| IV. | CONCLUSION..... | 107 |

LIST OF APPENDICES

- Appendix A *Curriculum Vitae* of Michael Franz, Ph.D.
- Appendix B Documents Cited

I. INTRODUCTION

1. I, Michael Franz, have been retained by Petitioners Netflix, Inc. (“Netflix”) and Roku, Inc. (“Roku”) (collectively, “Petitioners”) to investigate and opine on certain issues relating to United States Patent No. 8,407,609 (“the ’609 patent”) in their Petition for Inter Partes Review of that patent. The Petition requests that the Patent Trial and Appeal Board (“PTAB” or “Board”) review and cancel claims 1-3 of the ’609 patent.

2. The opinions set forth in this declaration are based on my personal knowledge, my professional judgment, and my analysis of the materials and information referenced in this declaration and its exhibits.

3. I am being compensated for consulting services including time spent testifying at any hearing that may be held. I am also reimbursed for reasonable and customary expenses associated with my work in this case. I receive no other forms of compensation related to this case. My compensation does not depend on the outcome of this inter partes review or the co-pending district court litigation, and I have no other financial interest in this inter partes review.

4. I understand that the ’609 patent has been assigned to Uniloc 2017 LLC.

5. This declaration is based on the information currently available to me. To the extent that additional information becomes available, I reserve the right to

continue my investigation and study, which may include a review of documents and information that may be produced, as well as testimony from depositions that have not yet been taken.

A. Qualifications

1. Education

6. I completed my undergraduate studies with a Diplomingenieur from the Swiss Federal Institute of Technology in Zurich (“ETH Zurich”) in 1989. In 1994, I obtained my Doctorate of Technical Sciences from ETH Zurich. My dissertation was entitled “Code-Generation On-the-Fly: A Key to Portable Software.”

2. Work Experience

7. I am a tenured Chancellor’s Professor of Computer Science in the Donald Bren School of Information and Computer Sciences at the University of California, Irvine (“UCI”). I am also, by courtesy, a Full Professor of Electrical Engineering and Computer Science in the Henry Samueli School of Engineering at UCI. In 2016, the University awarded me the title of distinction of “Chancellor’s Professor.”

8. I have served as a visiting professor at ETH Zurich, the University of California at Berkeley, the University of Klagenfurt in Austria, and the Technical

University of Berlin, the Technical University of Brunswick, and the University of Ulm in Germany.

9. I have been elevated to Fellow of the Institute of Electrical and Electronics Engineers (IEEE), the global engineering society. Fellow is the highest of three grades of membership that are awarded based on merit. In every year, IEEE limits the number of new Fellows to one tenth of one percent of the membership, which currently stands at about 430,000 members.

10. I have also been elevated to Fellow of the Association for Computing Machinery (ACM), the global professional society for computer scientists. Fellow is the highest of ACM's four grades of membership. ACM's rules for Fellows are even more restrictive than IEEE's, limiting the total number of Fellows in absolute terms to 1% of the membership, which currently stands at about 100,000 members. In recent years, ACM has typically elevated no more than 50 individuals to Fellow status in a single year; in 2015, the year I was advanced, there were 42 new Fellows.

11. Furthermore, I am a recipient of the IEEE Computer Society's Technical Achievement Award, "for pioneering contributions to just-in-time compilation and optimization and significantly advancing Web application technology." At most 5 of these awards are given annually by the IEEE Computer

Society, the largest of the IEEE's technical societies with a current membership of more than 60,000 members. I am also a recipient of the 2019 Humboldt Research Award, also known as the "Humboldt Prize." The award, given by the Alexander von Humboldt Foundation of Germany and funded by the German federal government, recognizes renowned researchers outside of Germany whose "fundamental discoveries, new theories or insights have had a significant impact on their own discipline and who are expected to continue producing cutting-edge achievements in the future." It is the highest award given by the Foundation to researchers based outside of Germany.

12. I have led pioneering research on downloadable code in client-server settings such as what we today call "Web 2.0." This research has had a real and lasting impact on a great many people. I am the co-inventor (with one of my former Ph.D. students) of the "Trace Tree" compilation technique, for which the United States Patent and Trademark Office has awarded U.S. Patent No. 8,769,511. I collaborated with the non-profit Mozilla Foundation to incorporate this technique into the Firefox web browser, where it became the basis of the "TraceMonkey" JavaScript engine, eventually used by several hundred million people every day.

13. Over the course of my career so far, I have been the Principal Investigator on several high-profile research projects with a total budget of almost

\$20M. My expertise in software systems with distinct emphases on performance and security of client-server and mobile computing has been sought out repeatedly by the Federal Government, and I have been participating in many high-level invitation-only meetings on Critical Infrastructure Protection and Cyber Security organized by the National Intelligence Community, the Department of Defense, the Department of Homeland Security, and the Department of Energy.

14. I am an Associate Editor of one the flagship journals of the IEEE, the IEEE Transactions on Dependable and Secure Computing (TDSC), and on the editorial board of two further peer-reviewed scholarly journals focusing on software engineering, Software – Practice and Experience (SPE) and Computer Science – Research and Development (CSRD). I have served on the program committees of most major academic conferences that are related to the various themes of my research. I have served as the primary advisor to 28 completed Ph.Ds. and currently serve as the primary advisor on 11 further dissertations in progress.

3. Publications

15. In addition to my dissertation, I co-authored “Automated Software Diversity,” released in 2015.

16. I have published 35 reviewed journal and magazine articles since 1993, and over 100 conference and workshop papers.

17. I am an inventor on five issued U.S. patents, as well as an additional patent that has been given a “notice of allowance” but that has not issued yet. I have one additional U.S. patent application pending.

4. Curriculum Vitae

18. A copy of my curriculum vitae is attached as Appendix A to this declaration.

B. Materials Reviewed

19. My opinions expressed in this declaration are based on documents and materials identified in this declaration, including the '609 patent, the prior art references and background materials discussed in this declaration, and the other references specifically identified in this declaration. I have considered these materials in their entirety, even if only portions are discussed here.

20. I have also relied on my own experience and expertise in Web technologies.

C. Level of Ordinary Skill in the Art

21. I am not an attorney and offer no legal opinions. I have been informed about certain aspects of the law for purposes of my analyses and opinions.¹

22. I understand that in analyzing questions of invalidity and infringement, the perspective of a person having ordinary skill in the art (“POSA”) is often implicated, and the Court may need assistance in determining that level of skill.

23. I understand that the claims and written description of a patent must be understood from the perspective of a POSA. I have been informed that the following factors may affect the level of skill of a POSA: (1) the educational level of the inventor; (2) the type of problems encountered in the art; (3) the prior-art solutions to those problems; (4) the rapidity with which innovations are made; (5) the sophistication of the technology; and (6) the educational level of active workers in the field. A person of ordinary skill in the art is also a person of ordinary creativity in the art.

¹ I understand that the patent laws were amended by the America Invents Act (AIA), but that the earlier statutory requirements still apply to pre-AIA patents. I have been informed that the ’609 Patent is a pre-AIA patent, so the pre-AIA requirements control. Unless otherwise stated, my understanding of the law about patent invalidity as set forth in this declaration relates to the pre-AIA requirements.

24. Based on my experience in client-server system design, as well as my reading of the '609 Patent, it is my opinion that a person of ordinary skill with respect to the subject matter of the '609 Patent at the time of the alleged priority date of the '609 Patent in 2008 would have had at least a B.S. degree in computer science, computer engineering, or electrical engineering (or equivalent experience) and would have had at least one year of experience with web development, including the then-current web technologies such as HTML, XML, Java, and JavaScript. This definition is flexible, and additional educational experience in computer science could make up for less work experience and vice versa.

25. I am at least a person of ordinary skill in the art and was so on the date to which the '609 Patent claims priority (August 21, 2008). As shown by my qualifications and my curriculum vitae attached as Appendix A, I am aware of the knowledge and skill possessed by a person of ordinary skill in the art at the time of the priority date of the '609 Patent. In performing my analysis, I have applied the standard set forth above.

D. Summary of Opinions

26. I have reviewed and analyzed the '609 Patent (Ex. B-1, same as Ex. 1001 in the Petition) as well as prior art references Davis (Ex. B-2, same as Ex. 1003 in the Petition), Choi (Ex. B-3, same as Ex. 1004 in Petition), and Siler (Ex. B-4, same as Ex. 1005 in Petition).

27. Based on my review and analysis, it is my opinion that claims 1-3 of the '609 Patent are invalid as obvious based on Davis in view of Choi. Based on my review and analysis, it is also my opinion that claims 1-3 of the '609 Patent are invalid as obvious based on Siler in view of Davis.

II. OVERVIEW OF THE TECHNOLOGY

A. Priority Date of the Claims

28. I have been informed that a U.S. patent application may claim the benefit of the filing date of an earlier patent application if the earlier patent application disclosed each limitation of the invention claimed in the later-filed U.S. patent application. I have also been informed that priority is determined on a claim-by-claim basis so that certain claims of a patent may be entitled to the priority date of an earlier-filed patent application even if other claims of the same patent are not entitled to that priority date.

29. I have also been informed that for patent applications filed before March 16, 2013, a patented claim is invalid if the claimed invention was patented or described in a printed publication in any country more than one year before the effective filing date of the claim, regardless of when the applicant conceived of the claimed invention.

30. I understand that the '609 Patent claims a priority date of August 21, 2008.

B. Overview of Relevant Technology When the '609 Patent Was Filed

1. Usage Tracking

31. In the realm of the Internet and World Wide Web, “usage tracking” is a term that is generally used to describe various techniques for tracking how users interact with content accessed over the internet. I will refer to this content as “Web resources” below, although it includes internet content well beyond simple HTML-based web pages, including multimedia content, downloadable “Applets” and scripts, etc. Usage tracking could include counting the number of visits to a webpage, tracking what parts of the webpage a user clicks on, and many other possible methods. While there are many different activities that might be tracked, and many different approaches to performing the tracking, what is thematic of all types of usage tracking is that the owner, operator, administrator, etc. of some Web resource wants to see how users interact with the Web resource. But because the user is located remotely and is only accessible through the Internet and World Wide Web, the owner, operator, administrator, etc. may have to channel the tracking through the Web resource itself.

32. My reference to owners, operators, administrators, etc. above reflects the fact that usage tracking has historically been employed by many different types of actors. Amateur website publishers often track the number of visits to their

webpages, which can be implemented entirely server-side. Website administrators might track not just the number of visits, but how the volume of those visits vary throughout the day. Other actors have used more sophisticated usage tracking methods, some of which cannot be implemented solely on the server but require client-side support.

33. While many different types of actors have employed usage tracking, the advancement in usage tracking technologies has typically been driven by actors with some direct monetary interest. It is perhaps not surprising that actors that are paying or being paid based on user interaction with Web resources have had significant interest in increasing the accuracy of usage tracking and/or the types of usage tracking over time.

34. On the Web, the largest class of actors with a direct monetary interest consists of advertisers. Advertising money in fact has been driving many facets of the Web and its development over time, and is driving some of the largest corporations in the Internet economy, such as Google and Facebook. Relating back to my point above about the motivations to increase usage tracking breadth and accuracy, advertisers have been very keen to make advancements in usage tracking since the very beginning of the Web. At one point, it was common for advertisers to pay a fee based on the number of clicks of their ads (e.g., pay a fee to

a webpage owner when a user clicks on a banner ad on that webpage). But this is something of a rudimentary measure of advertising value, so advertisers have sought better usage tracking metrics to value how much they should pay.

35. I now give an explanation of some of the developments in usage tracking over time.

36. The World Wide Web became publicly available in 1991. Webpages were fairly simple at that time, and usage tracking was also pretty rudimentary. In the early and mid-1990s, usage tracking included things like tracking the number of times a webpage was visited, and tracking the number of times a link was clicked. These metrics were fairly easy to implement, even in very simple webpages.

37. By the late-1990s to the early-2000s, webpages had advanced in functionality and complexity, and usage tracking advanced along with them. One significant part of these advancements was the proliferation of downloadable executable “Applets,” written in languages such as Java, and downloadable “Scripts,” written in languages such as JavaScript. Both of these allowed more dynamic functionality on the client side. Java and JavaScript first appeared in 1995 and would ultimately become very popular. With these solutions to providing “executable content” to a user’s browser it was also possible to perform

more advanced usage tracking by actually following users' actions on the users' computers themselves. So for example, it became possible to determine (by an Applet or a script running in the background) whether an advertisement was currently visible on the screen, or whether it might be temporarily scrolled off-screen or obscured by another window.

38. By 2008 (the '609 Patent's claimed priority date), Web technology had advanced even further. Webpages were highly functional with dynamic content, streaming media, and other features that are quite similar to Web technology today. Usage tracking advanced in correspondence, including some very fine-grained techniques, such as tracking a user's passive interest in various parts of a webpage by tracking where the user's cursor hovers, how long it hovers, etc.

39. U.S. Patent Application Publication 2002/0165849, Ex. B-10 ("'849 Publication"), which was published in 2002 provided some background on the state of the art in usage tracking and its motivations from the advertising community in the early 2000s. The '849 Publication explained that the Internet had transformed into a "global marketplace" driven by the use webpages through the World Wide Web. '849 Publication, ¶0005. The webpages were formatted using HTML and included multimedia such as "graphics, audio, and moving

pictures.” ’849 Publication, ¶0006. The Web was an “increasing attractive medium for advertising and other business purposes.” ’849 Publication, ¶0006. “The Internet has emerged as an attractive new medium for advertisers of information, products and services to reach consumers.” ’849 Publication, ¶0008.

40. In the early 2000s, advertisers typically had to track various metrics to determine whether the money paid for an advertising listing had been financially worthwhile. ’849 Publication, ¶0016-0026. Advertisers would “keep track of the number of clicks that a listing is getting.” ’849 Publication, ¶0021. Advertisers would track the “click through rate” for a listing. ’849 Publication, ¶0022.

41. But needs remained in the state of the art advertising technology. These existing approaches were difficult for advertisers to manage. ’849 Publication, ¶0016-0026. And then-common pricing schemes for advertising were not ideal. ’849 Publication, ¶0012-0015. Advertisers often paid for each impression for an advertisement (e.g., for each visit to the webpage displaying a banner advertisement). ’849 Publication, ¶0012. But click through rates were low, and many of the impressions were for users not interested in the advertised product or service. ’849 Publication, ¶0013.

42. U.S. Patent 6,108,637, Ex. B-11 (“’637 Patent”), which issued in 2000, is an example of how practitioners in the field were trying to improve usage

tracking techniques as a way to address these shortcomings in the field. The '637 Patent stated that “advertisers have particular interest in knowing how and to what extent their advertisements are displayed and/or observed, since such knowledge can be a key element in evaluating the effectiveness of their advertising and can also be the basis for payment for advertising.” '637 Patent, 1:35-51. The '637 Patent also noted the deficiencies of the then-standard usage tracking (“monitoring” in that reference) approaches. '637 Patent, 1:52-65. The '637 Patent sought to address the problems by improving usage tracking on the client itself, by using a client-side tracking program provided by a server and downloaded to the client. At the time of the invention, there were generally two well-known technologies for such “executable content,” programs downloaded from a server to a client and then executed on the client. The first of these are so-called “Applets” that are typically represented as programs for the Java Virtual Machine (JVM) and that are executed by a JVM that is part of the user’s browser. The other is so called “scripts,” with the JavaScript language being the most popular browser scripting language. The most important difference between Applets and Scripts is that Applets are shipped as compiled code (in the case of the JVM, typically compiled from the Java source language) and are therefore somewhat harder to reverse engineer, while scripts are shipped in human-readable

source code form (but today they often are also heavily obfuscated). Hence, an Applet downloaded to the client could track “whether (and for how long) the content display is hidden” and “whether the content display is fully hidden or partially hidden (and for how long the content display is fully and partially hidden, respectively).” ’637 Patent, 7:4-30, 11:57-12:43.

43. U.S. Patent 7,310,609, Ex. B-14 (“’0609 Patent”), which was filed in 2002, demonstrated other ways in which usage tracking was being improved for the purposes of advertising. The ’0609 Patent explained that metrics like “the number of times that the Web page containing the advertisement is displayed” were used “by the providers of such services and advertisers, typically in order to calculate advertising rates.” ’0609 Patent, 1:61-65. The ’0609 Patent also mentioned advertisement click-throughs as a metric used by advertisers. ’0609 Patent, 1:66-2:13. “Advertisers, however, would like not only to count a number of ‘impressions,’ or how many times their advertisement is seen, but also to find a way to track how effective their ads are in attracting consumers’ interest in their products.” ’0609 Patent, 2:9-13.

44. The ’0609 Patent provided a solution to address these needs of advertisers. The ’0609 Patent disclosed providing a web browser Applet to the client computer. ’0609 Patent, 2:24-3:5. The Applet would track “how long an

object is displayed, which objects are selected by a user, which items are considered by a user according to the amount of time the cursor hovers over the items, measuring the time of presentation of an element in various ways, and/or activating hyperlinks.” ’0609 Patent, 2:24-3:5.

45. U.S. Patent 7,089,304, Ex. B-12 (“’304 Patent”), which was filed in 2001, demonstrates that advancements in usage tracking were also being made by groups other than advertisers. The ’304 Patent disclosed that, according to the ways that clients were charged for Internet access at the time, Internet service providers were “traditionally responsible for tracking a client’s usage, if necessary.” ’304 Patent, 1:38-47. The Internet service provider might track how long a client accessed the Internet, as well as what services the client accessed. ’304 Patent, 1:48-2:22. The ’304 Patent disclosed use of a type of “metering packet” to make this tracking more effective. ’304 Patent, 2:26-3:56.

46. U.S. Patent 6,877,007, B-13 (“’007 Patent”), filed in 2001, demonstrates that highly detailed usage tracking information was of considerable interest as a general matter for any business operating on the Web. “It is a truth universally acknowledge that, in order to succeed, a business must study the habits, desires, and behavior of its customers. For companies conducting business over the Internet and the World Wide Web ... this necessarily extends to examining and

measuring their customers' interaction with their Web sites." '007 Patent, 1:14-19.

The '007 Patent disclosed that this goal could be achieved by providing a tracking program with a webpage and having the tracking program track the user's interaction with the webpage on the client. '007 Patent, 1:50-2:28. The user's interactions are transmitted to a server and can then be later replayed. '007 Patent, 1:50-2:28.

47. While much more could be said about the multitude of usage tracking techniques that had been developed by 2008, the techniques highlighted above give a good overview of how the technology has advanced over time. These examples also demonstrate that by 2008, tracking how long users interacted in various ways with various types of content in their web browser were very well-known and easily implementable by any person of ordinary skill in the field.

48. I use the term "usage tracking" in this declaration, but I note that other terminology may be used to refer to this domain. I have seen terms like "user tracking," "interaction tracking," "user analytics," and many others. These other terms also refer to this general field of tracking the usage of Web resources.

2. Streaming Media

49. The use of streaming media in the World Wide Web was also very well-developed by 2008. While early webpages tended to be static and simple and

contain primarily static images or multimedia, this was already changing by the late-1990s.

50. One way in which streaming media was advancing by the late-1990s was through standardization processes, with the Internet Engineering Task Force (IETF) leading some of these efforts. In 1996, the IETF released RFC 1889, which described Real-time Transport Protocol (RTP), a protocol for performing real-time delivery of content. The “real-time” part of the delivery in RTP was significant, because it meant that content could potentially be transferred as it was created and also that content could potentially be presented as soon as just part of it was received. RTP did not describe those specifics, because it was a network transport protocol. But it provided the groundwork on which such applications could be built.

51. The IETF released complementary specifications with RTP and in the years the followed. RFC 1889 itself described Real-time Transport Control Protocol (RTCP), which provided a control layer over the delivery functions handled by RTP. In 1998, the IETF released RFC 2326, which described Real-Time Streaming Protocol, which was intended as a protocol for real-time streaming like RTP and RTCP, but for providing complimentary “network remote control” functionality. RTP was also itself re-released as RFC 3550 in 2003.

52. Commercial efforts to advance streaming media were also taking place in the late-1990s. RealNetworks was one of the major early players in this field and had stable audio streaming technology in commercial use by the end of the 1990s.

53. Throughout the 2000s and up through 2008, streaming media over the Internet expanded at an incredible rate. By August 2008, YouTube had been live for several years and was streaming millions of videos every day. Netflix and Hulu were offering video streaming services. Streaming audio and video were commonplace and an increasingly preferred method for delivering audiovisual content over the Web to users.

54. By August 2008, other complimentary technologies had also developed related to streaming media. One such technology was the content delivery network (CDN). CDNs had existed as commercial entities since at least the late-1990s, when Akamai Technologies was formed. The purpose of CDNs was and is to reduce latency by storing content “closer” to the end-users. Typically, CDNs such as Akamai operate so called “edge caches” in many major cities, and as the name suggests, these servers cache frequently accessed content. User requests that would normally go to a content provider’s “main” server(s) are

redirected to these edge-caches instead, typically resulting in a much quicker response time because the cache is fewer “hops” away from the user.

55. Using CDNs makes particularly much sense for large media content such as videos, both for a downloading and for a streaming context. In the case that a video needs to be downloaded fully before it can be played, a video server that is “far away” from a client (in terms of router hops from the server) might lead to a long start delay (“high latency”) since the complete video file needs to be transported to the client first. Such a file would be transmitted as a sequence of separate packets, and the more “hops” there are between the server and the client, and the longer the file, the more chances exist that a packet can get lost and needs to be retransmitted.

56. In the case of streaming video, a “far away” server might lead to interruption of the video stream if packets get lost or delayed due to retransmission, or if the buffer size is increased to account for this, a resulting startup delay (high latency). By placing CDNs “close” to the users, the latency can be reduced significantly.

57. Through these industry standards, common commercial exploitation, and supporting technologies, it is possible to see how streaming media, especially streaming video, had become very well-known by 2008, and something that a

person of ordinary skill in this field could easily implement without significant challenges.

C. The '609 Patent

58. The '609 Patent describes a type of webpage through which content can be uploaded, browsed, and viewed. '609 Patent, Figure 2. The disclosure is made with respect to standard networking hardware, such as servers and personal computers. '609 Patent, Figure 1. The webpage includes a content search feature. '609 Patent, Figure 3. The webpage allows a user to upload content. '609 Patent, Figure 4. The webpage allows a user to associate information with content that has been uploaded elsewhere. '609 Patent, Figure 5. The webpage allows a user to create content, such as an audio show. '609 Patent, Figures 6-7. The webpage allows for automated association of information with content that has been uploaded elsewhere. '609 Patent, Figure 8. The webpage allows the content itself to be displayed. '609 Patent, Figure 9. The webpage allows tracking information to be sent periodically from a client displaying the webpage to a server. '609 Patent, Figure 10.

59. The claims in the '609 Patent deal with a subset of this disclosure, namely the part about tracking information. This is described primarily at column 7, lines 15 through 58, column 11, line 37 through column 14, line 8, and in Figures 9 and 10.

60. In column 7, the '609 Patent describes tracking various information for a webpage. The '609 Patent describes tracking: the number of visitors to a webpage, a number of visitors for a specific element of content, the number of pages viewed by a particular user, when a user began watching a presentation of content, when a user ended watching a presentation of content, and the total time the user spent on the webpage. '609 Patent, 7:15-58.

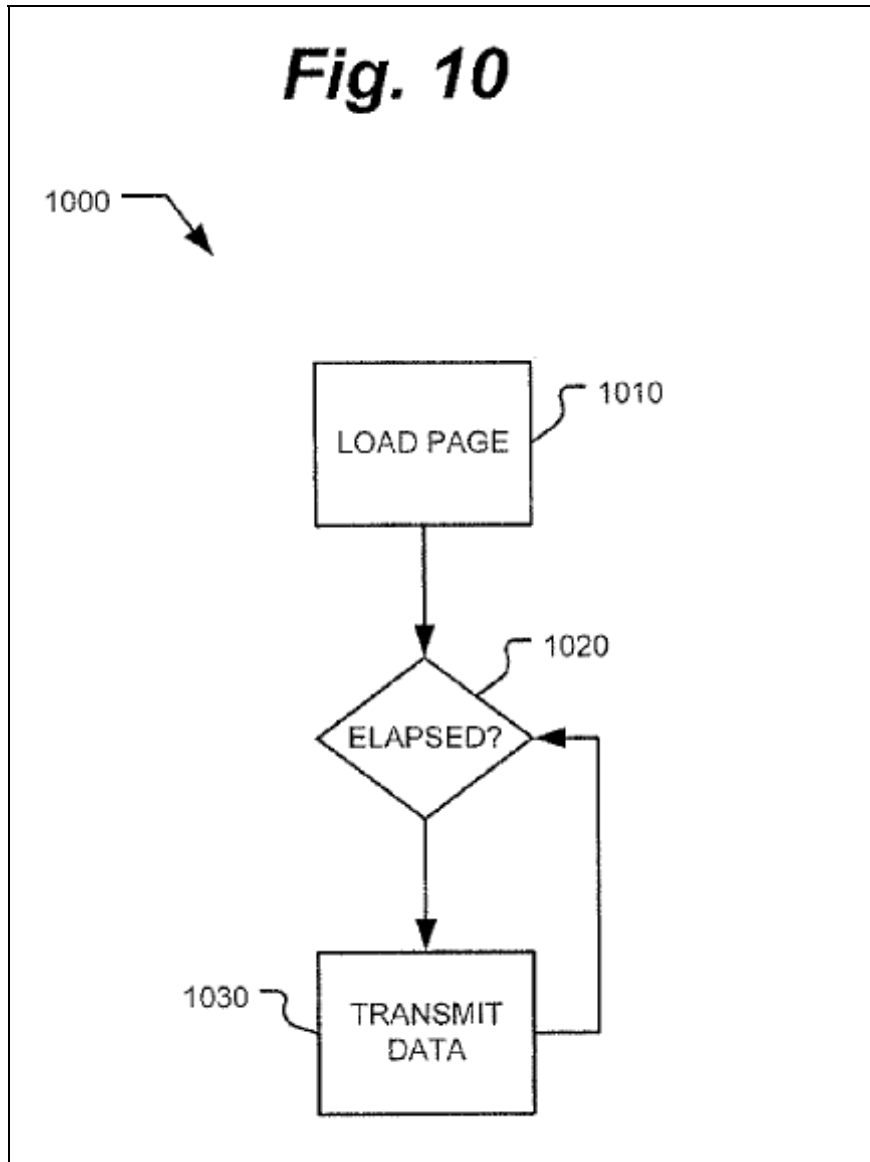
61. The '609 Patent repeats some of this disclosure in column 11. The '609 Patent describes tracking the number of accesses to content, as well as how long a user actually watches a presentation of content. '609 Patent, 11:37-58.

62. With respect to Figure 9, the '609 Patent discloses a webpage for displaying content and related information. The webpage can include a portion for displaying the actual content (see references to element 920), as well as portions for displaying advertisements (see references to element 910 and 940). '609 Patent, 11:59-12:15. The content can be streamed or downloaded. '609 Patent, 12:1-5.

63. The '609 Patent describes two different scenarios for which tracking information can be generated. If the content is uploaded to the same system that hosts the webpage, then the tracking information can be gathered on the server. '609 Patent, 12:16-35. When the content is uploaded to a different system than the one that hosts the webpage, it can be harder to gather the tracking

information. '609 Patent, 12:36-55, given that the server hosting the webpage does not have control over the content.

64. In order to have an approach that works regardless of where the content is hosted, the '609 Patent has a technique for gathering tracking information at a client. '609 Patent, 12:56-14:8. One element of this technique is to have the client periodically send tracking information back to the server that hosts the webpage, as depicted in Figure 10, reproduced below. '609 Patent, 12:56-13:23. And in order to provide this functionality in the client, an Applet with the timer functionality can be provided from the server hosting the webpage to the client. '609 Patent, 12:56-13:9. The server can then store the received tracking information. '609 Patent, 13:24-42.



65. The '609 Patent says that knowledge of how long a user spent on a webpage was not generally available at the time of the invention, '609 Patent, 13:43-48, even though this was actually quite well established as explained in the description of the history of the technology in this declaration, above. As for motivation for its disclosed usage tracking, the '609 Patent explains that the

technique allows improvement in the “scale of payments for advertising displayed” on the webpage. ’609 Patent, 13:48-14:2.

D. The Challenged Claims

66. I understand that the Petitioners are challenging all of the claims in the ’609 Patent, which means claims 1-3. I reproduce those claims below, with small letters added to enumerate the elements of claim 1.

1. A method for tracking digital media presentations delivered from a first computer system to a user’s computer via a network comprising:

[a] providing a corresponding web page to the user’s computer for each digital media presentation to be delivered using the first computer system;

[b] providing identifier data to the user’s computer using the first computer system;

[c] providing an applet to the user’s computer for each digital media presentation to be delivered using the first computer system, wherein the applet is operative by the user’s computer as a timer;

[d] receiving at least a portion of the identifier data from the user’s computer responsively to the timer applet each time a predetermined temporal period elapses using the first computer system; and

[e] storing data indicative of the received at least portion of the identifier data using the first computer system;

[f] wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from the first computer

system directly to the user's computer independent of the first computer system;

[g] wherein the stored data is indicative of an amount of time the digital media presentation data is streamed from the second computer system to the user's computer; and

[h] wherein each stored data is together indicative of a cumulative time the corresponding web page was displayed by the user's computer.

2. The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.

3. The method of claim 2, wherein the received data is indicative of a temporal cycle passing.

E. Claim Construction

67. I understand that claim terms generally are construed in accordance with the ordinary and customary meaning they would have to a POSA at the time of the invention in light of the claim language, the specification, and the prosecution history. I understand that dictionaries and other extrinsic evidence may be considered as well, though such evidence is typically regarded as less significant than the intrinsic record in determining the meaning of the claim language

68. For all terms of the challenged claims of the '609 patent, I have interpreted them as they would have been understood by a POSA at the time of the invention, *i.e.*, August 21, 2008.

69. It is my opinion that the following terms require construction to address this Petition; I do not address terms that may require construction in the district court.

1. “Computer System”

70. I agree that the best construction for “computer system” in claim 1 is “a single computing device or a collection of computing devices having a common operator or under common control.”

71. I believe this is the most appropriate construction because that is nearly verbatim the definition given in the '609 Patent itself. '609 Patent, 3:52-55. This definition also comports with how the term “computer system” is used in claim 1, so I do not see any reason to deviate from the definition given in the specification. In particular, claim 1 refers to a “first computing system” and a “second computing system,” with the two “distinct” from one another. '609 Patent, 14:35-39. This seems to match the third-party hosting of content, as described in the specification. '609 Patent, 12:36-45. And the definition given in the specification clarifies what is different between the two systems.

72. For these reasons, it is my opinion that “computer system” should be construed as “a single computing device or a collection of computing devices having a common operator or under common control.”

2. “Streamed”

73. I agree that the best construction for “streamed” is “transferred via a technique such that the data can be processed as a substantially steady or continuous sequence.”

74. I believe this is the most appropriate construction because that is very close to the definition given in the ’609 Patent itself. ’609 Patent, 4:43-47. This definition also does not conflict with how the term “streamed” is used in the claims, so I do not see any reason to deviate from the definition given in the specification.

75. The specification defines “streaming” as a way to contrast with “downloading,” which the specification also defines. ’609 Patent, 4:40-52. The specification explains that “streaming” is a technique for transferring data in such a way that it can be processed as a substantially steady or continuous stream, and so that presentation of the data can be started before all of the data has been transferred. ’609 Patent, 4:43-47. The specification defines “downloading” as transmitting data, namely “an entire data file,” between computers. ’609 Patent, 4:48-52. Thus the specification describes both “streaming” and “downloading” as different types of data transfer.

76. The language of claim 1 also suggests that “streamed” is a type of data transfer. Namely, claim 1 recites that data is streamed “**from** a second computer system” and “**to** the user’s computer.” ’609 Patent, 14:35-39. This thus explains a way to transfer data between computers.

77. I understand that another petitioner has requested inter partes review of the ’609 Patent, and that that petitioner proposed a construction of “streamed” that focuses on the presentation of content. While “streamed” can be used colloquially to refer to presentation of content (e.g., “I streamed a movie last night.”), I do not believe that is the correct construction of “streamed” for the reasons described above. Nonetheless, I have been asked to also consider the claims of the ’609 Patent and the prior art in light of a “presentation” type of streaming as an alternative approach, and I do so in my analysis below.

III. UNPATENTABILITY OF THE ’609 PATENT CLAIMS

A. Standards for Invalidity

1. Obviousness

78. I am informed and understand that a patent cannot be properly granted for subject matter that would have been obvious to a person of ordinary skill in the art at the time of the alleged invention, and that a patent claim directed to such obvious subject matter is invalid under 35 U.S.C. § 103. It is also my understanding that in assessing the obviousness of claimed subject matter, one

should evaluate obviousness in light of the prior art from the perspective of a person having ordinary skill in the art at the time the alleged invention was made (and not from the perspective of either a layman or a genius in that art). It is my further understanding that the question of obviousness is to be determined based on:

- The scope and content of the prior art;
- The difference or differences between the subject matter of the claim and the prior art (whereby in assessing the possibility of obviousness one should consider the manner in which a patentee and/or a Court has construed the scope of a claim);
- The level of ordinary skill in the art at the time of the alleged invention of the subject matter of the claim; and
- Any relevant objective factors (the “secondary indicia”) indicating nonobviousness, including evidence of any of the following: commercial success of the products or methods covered by the patent claims; a long-felt need for the alleged invention; failed attempts by others to make the alleged invention; copying of the alleged invention by others in the field; unexpected results achieved by the alleged invention; praise of the alleged invention by the alleged infringer or others in the field; the taking of licenses under the patent by others and the nature of those licenses; expressions of surprise by experts and those skilled in the art at the subject matter of the claim; and whether the patentee proceeded contrary to accepted wisdom of the prior art.
- Any relevant objective factors (the “secondary indicia”) indicating obviousness: independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it; and other evidence tending to show obviousness.

B. Ground I: Claims 1-3 were obvious in view of Davis and Choi

1. The Davis-Choi Combination

a. Davis

79. Davis disclosed a way to perform usage tracking, and in particular a way to track the number of visits to a webpage and the duration of time that a webpage is displayed to a user. Davis discloses its techniques in several variations, but I focus my analysis on the technique described in Figure 4, which is also described as an extrapolation of Figure 3. Davis, 11:34-13:18, 9:16-11:33. Figure 4 is shown below, with markup added in the form of coloring, shading, and the numerals 1 through 6.

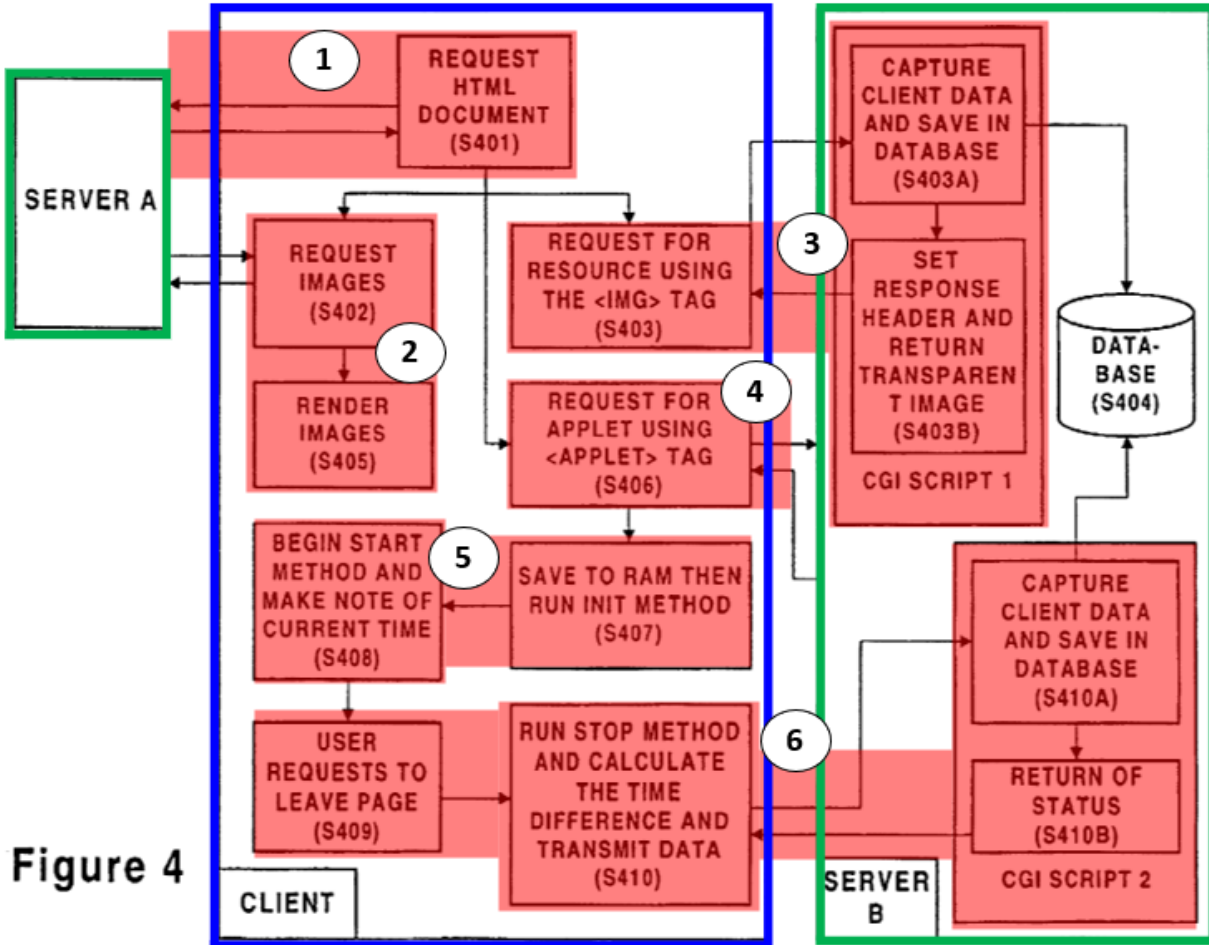


Figure 4

80. I now summarize the disclosure from Davis for Figure 4 with reference to the marked-up figure above.

81. The system of Figure 4 includes a Client, Server A, and Server B. The client is a typical user's computer. Davis, 5:14-56, 6:1-4. The Server A provides the webpage displayed to a user, and may host some of the media content that is displayed in the webpage. Davis, 11:34-12:4. A third-party server may also provide the media content that is displayed in the webpage. Davis, 11:34-12:4. The Server B provides the tracking functionality, includes a database to store

tracking information, and several scripts that cause the tracking functionality to work. Davis, 11:34-12:50.

82. At the step (1), Server A provides a webpage to the Client. Davis, 11:35-40. This is a standard webpage, such as one using HTML. Davis, 11:34-12:4. The webpage is provided using standard Internet technology, such as TCP/IP and HTTP. Davis, 11:34-12:4.

83. At the step (2), the Client renders images or other content that is embedded in the webpage. Davis, 11:35-47. The image and other content can also be provided by the Server A, or they can be provided by some other system. Davis, 11:34-12:4. In this sort of arrangement, the Client would determine what image or other content is embedded based on some reference (e.g., a URL) included in the HTML of the webpage.

84. At the step (3), the Client uses a link that is embedded in the webpage to invoke CGI Script 1 that is available on Server B. Davis, 11:47-12:4. The purpose of CGI Script 1 is to allow the Server B to register the Client. Davis, 11:34-12:4. This will then allow the Server B to associate the tracking information later received (through CGI Script 2) with a particular Client device. Davis, 11:34-12:50. Davis explains that this registration and setup can be performed using a

cookie, which the Server B can create and provide to the Client. Davis, 11:34-12:50.

85. At the step (4), the Client obtains an Applet from the Server B. Davis, 12:13-21. The Applet is “executable content,” i.e. a mobile program that is executed on the client and that contains the tracking functionality. Davis, 12:5-50. An Applet was a well-known functionality at the time of Davis’s disclosure. An Applet was typically implemented in web browsers as compiled code targeting the Java Virtual Machine (JVM) that was present (and still is) in most browsers at the time. The JVM takes the Applet and executes it step by step, similar to how a microprocessor would execute a program binary instruction by instruction.

86. At the step (5), the Client runs the Applet. Davis, 12:22-26. The Client runs the Applet in coincidence with the webpage display beginning. Davis, 12:5-50. This will allow the Applet to later determine how long the webpage was displayed, based on the functionality performed at step (6). Davis, 12:5-50. The Applet is thus a sort of timer embodied in an Applet. Davis, 12:5-50.

87. At the step (6), the Applet stops executing when the user leaves the webpage. Davis, 12:26-40. The Applet can then calculate the amount of time the webpage was displayed by comparing the time that it began execution with the time that it ended execution. Davis, 12:5-50. The Client then invokes the CGI

Script 2, which involves providing the tracking information generated by the Applet at the Client. Davis, 12:5-50. The Server B then stores that tracking information in a database. Davis, 12:5-50.

88. Similar to the '609 Patent and much of the advancements in usage tracking (see description of the technology background above), Davis explained its motivation as one of monetization. Davis, 11:24-33. In particular, Davis's approach could allow better monetization of advertisements displayed on a webpage. Davis, 11:24-33.

b. Choi

89. Choi disclosed an approach for providing streaming media, and in particular for recovering from network disruptions that occur while providing the streaming media. Choi explained that streaming media over networks posed technical challenges, such as network congestion, packet loss, variable latencies, and others. Choi, ¶¶0002-0004. Choi described a way to deal with these disruptions, which dealt largely with a new type of message, a "reconnect request," that a client could send to a server to reestablish the streaming media connection. Choi, ¶¶0005-0014. Choi explained that this approach could be implemented with industry-standard technologies like RTSP, HTTP, and others, including those that I introduced elsewhere in this declaration. Choi, ¶¶0006, 0029-0031.

90. Choi explained with respect to Figure 3 that the client and server could establish a session in which the stream could be transmitted. Choi, Figure 3, ¶0044. A session identifier could be used to identify the session between the client and server, and a stream identifier could be used to identify the stream of content being provided from the server to the client. Choi, ¶0044. Choi here is using “session” to refer to a period of interaction between a specific client and a specific server. This is similar to the way that Davis uses the CGI Script 1 to start a period of interaction between the Client and Server B.

91. One feature of Choi’s system is a periodic reporting message that the client sends to the server. Choi, ¶0047. Choi calls these “state data” or “logging statistics.” Choi, ¶0047. The data reported from the client to the server allows an administrator to see what each client is doing at any given time. Choi, ¶0047. The reporting message includes the logging statistics along with the session identifier and the stream identifier. Choi, ¶0047. The server can later use the logging statistics that it stores to help reestablish the connection with the client after a disruption. Choi, ¶0047-0051.

92. Choi explains that the particular logging statistics that are provided are in Appendix C of Choi. Choi, ¶0049. Appendix C describes the general reporting message interaction between the client and the server. Choi, ¶0096-0099.

Choi explains that the interval at which the client will send the reporting messages is determined based on a “reporting interval parameter” set when in one of the first messages between the client and the server.

93. Appendix C then includes a Table C1 labeled as “Logging Fields” that identifies the fields of data reported from the client to the server and stored by the server. Choi, Table C1. The parameters include a “cs-uri-stream” parameter, which identifies the “URI stem of the content that was requested.” Choi, Table C1. The parameters include a “c-starttime” parameter, which identifies the timestamp “when you played the file in normal mode.” Choi, Table C1. The parameters include a “c-endtime” parameter, which identifies the timestamp “when the Client finished playing the file in normal mode.” Choi, Table C1. The parameters include an “x-duration” parameter, which identifies “How long you tried to receive the stream (in seconds).” Choi, Table C1. Many other parameters are included in Table C1.

c. Motivation to Combine

94. Based on my review of Davis and Choi, and based on the background of the field in August 2008, it is my opinion that a POSA would have been motivated to, and would have found it obvious to, combine several features described in Choi into the system described in Davis.

95. First, Davis may not directly and explicitly be describing streaming content in the webpage provided by Server A. When describing the technique shown in Figure 4, Davis describes images in the webpage, and that the images are “fetched.” Davis, 11:34-12:5. In this context, fetching images likely is closer to “downloading” of the images, rather than “streaming” them.

96. At the same time, Davis explains that other type of content, including audio and video, can be presented in the webpage. Davis, 7:1-29. So a person of ordinary skill would have understood that the fetching of images described in column 11 was one type of resource that could be presented in the webpage, but not the only type. The person of ordinary skill would have understood that audio and video were equally as possible to be displayed in the webpage.

97. In situations where audio and video were provided in the webpage, Davis does not explicitly say that the content is “streamed” to the webpage. But a person of ordinary skill would have recognized this as a possibility even by the disclosure of Davis itself. Namely, Davis discloses a variation on its system where the content is a “live news or entertainment feed.” Davis, 16:63-17:10. As discussed previously, the use of “live” or “real-time” delivery of continuous content in the late-1990s through to August 2008 generally implied streaming of such content. Otherwise, if the user had to fully download the content at once

ahead of presenting it to the user, it could not realistically be called “live.” Hence, based on this “live feed” portion of Davis, a person of ordinary skill would have understood that the content presented in the webpage described with respect to Figure 4 could also have been streamed to the client.

98. In any case, even ignoring the suggestions of content streaming of Davis, a person of ordinary skill would certainly have recognized that streaming of content was a well known method for content delivery in 2008 and was described in many publications, including the disclosure of Choi. Choi is itself built entirely around the streaming of content. Choi, ¶0002-0014. Choi not only discloses that streaming of audio and video was possible, it goes a step further to improve streaming delivery of content. Choi, ¶0002-0014.

99. By the time Choi was filed in June 2002 and published in December 2003, streaming of audio and video was already extremely well known, as discussed previously in this declaration. This is implicit in Choi’s own disclosure. Choi, ¶0002-0014. And this is further evidenced by the fact that such standards as RTP and RTSP had been described by the IETF some five to ten years earlier, as discussed elsewhere in this declaration. In fact, Choi is really just a representative instance showing that streaming of content was well known, if that was not already evident from Davis.

100. And a person of ordinary skill as of August 2008 would not only have recognized the possibility of using streaming content in the webpage described with respect to Figure 3 of Davis, a person of ordinary skill would have been highly motivated to actually use streaming. First of all, based on the disclosure of Davis itself, a person of ordinary skill would have recognized that streaming would be the only viable delivery mechanism in cases where continuous live or real-time content was being provided, as discussed above. Davis, 16:63-17:10.

101. But even for non-live and non-real-time content, streaming had by August 2008 become the preferred delivery mechanism for audio and video in many user-facing Web applications. The reason is because by August 2008, lengthy audio and video content was being presented in webpages, not just short clips. Recall the discussion earlier in this declaration regarding the existence of video streaming services like Hulu and Netflix by August 2008. And it was quite evident that a user who wanted to view a movie or even a long video on YouTube did not want to have to start a download and then come back several minutes or hours later to watch it. Essentially the point of services like YouTube, Netflix, and Hulu was to deliver video content when the user wanted to watch it—cut out the delay. Additionally, users typically do not watch every video that they start all the way to the end – so downloading a complete video only to have the user later skip

over part of the already downloaded file is a colossal waste of network resources. So streaming was, by August 2008, what end users preferred and even expected. In this context, a person of ordinary skill would have been highly motivated to stream the audio or video content presented in the webpage provided from Server A to the Client in Davis's system.

102. A person of ordinary skill in August 2008 would also not have found it challenging to implement streaming audio or video as a delivery mechanism. As described previously in this declaration, by August 2008, streaming protocols had been standardized for a decade, streaming was used in many commercial implementations, and Davis's system disclosed that it used the standard Web technologies that a person of ordinary skill would have been familiar with. Davis, 7:1-29. There were also commercial ready-made components such as Adobe Flash available almost ubiquitously to make implementation of streaming solutions simpler. So a person of ordinary skill would have recognized the option of using streaming content in Davis's webpage, would have been motivated to use that delivery mechanism, and would have found it well within his/her skillset to put together an implementation using available conventional technologies.

103. A second feature disclosed in Choi that a person of ordinary skill would have been motivated to add, and would have found obvious to add, into

Davis's system was the use of a periodic timer to send the tracking information to the server. As described above, Davis uses a timer Applet that starts running when a webpage is first loaded, then stops running when the webpage is no longer displayed. Davis, 12:5-50. It is at this point that the Client sends tracking information to Server B. Davis, 12:5-50.

104. But this timing approach described in Davis was just one way to schedule the tracking information reporting messages. A person of ordinary skill in August 2008 would definitely have recognized several other possible approaches. One approach would have been to use batch processing. For instance, the Client could store a series of tracking information throughout the day, and then send a single message to the Server B at night.

105. Another approach would have been to use a regular, periodic reporting interval. This approach is the one described by Choi. Choi, ¶¶0047, 0097. A person of ordinary skill would have been familiar with this regular, periodic reporting interval. And a person of ordinary skill would have had no technical difficulty in implementing such an approach. For instance, Choi explained that the regular, periodic reporting interval could be used with industry standards like RTSP. Choi, ¶¶0006, 0029, 0047. A person having ordinary skill in the field as of

August 2008 would have recognized several significant benefits with using the regular reporting approach.

106. As a first reason, Choi discloses that the regular receipt of logging statistics at the server would make sure logging information was on hand in case any network disruptions occurred. Choi, ¶0047-0051. This contrasts with the batch approach or even the approach described in Davis. For example, with the approach described in Davis, if the user remains on the webpage for a long period of time, and there is a network disruption during that time, then there is an increased risk that the tracking information will be lost (e.g., Client observes lost connection to Server B and thereby does not invoke CGI Script 2 when the webpage is exited). The longer the time between tracking information reports, the more likely that something will go wrong.

107. As a second reason, using a predetermined reporting interval would in some instances be easier to implement than the event-based approach described in Davis. While Davis's timer is described as being quite simple, a predetermined reporting interval was a known technique which was in some ways even simpler. For a predetermined reporting interval, the implementing code would include basically two parts: (1) the logic to gather, package, and transmit the tracking information; and (2) the logic to wait for the predetermined reporting interval. The

latter part, element (2) as just annotated, is typically just one, two, or three lines of code in most programming languages. For instance, there is often a “sleep()” function or something similar in most programming libraries. This is incredibly simple, and one that basically any college student learning Web-based programming would be familiar with. A simple, familiar solution would be appealing to a person of ordinary skill in the field, or really any person in the field.

108. As a third reason, a person of ordinary skill in the field as of August 2008 would have been motivated to implement the predetermined reporting interval because it was one of basically only two or three potential timing approaches. As described above, the predetermined timing approach could have been used, as could have a batch reporting approach or an event-driven approach as in Davis. Of these three, the batch driven approach would not have been appealing to a person of ordinary skill in the field. As of August 2008, transactions and reporting over the Web were increasingly real-time, and batch reporting would have been considered somewhat counter to this trend. Also, batch reporting would have greatly increased the risk of delaying or even losing information (e.g., Client machine crashes, Client machine loses network connection to Server B, or user explicitly turns off the machine while closing credits are still running to avoid being charged for a movie).

109. So a person of ordinary skill would have viewed there as being basically two simple approaches for implementing the reporting: event-driven reporting, or a predetermined interval for reporting. Given these limited options, a person of ordinary skill would have been motivated to try either and both. A Web programmer would likely try one, perhaps try the other, and see which worked best. These would not have been difficult schemes to implement. And to the extent that such a programmer for some reason only wanted to spend time trying one approach, he/she would have likely implemented the predetermined interval for reporting for the reasons described above. This is all the more so given that the regular, periodic reporting approach would be recognizable to a person of ordinary skill as similar to the well-known “heartbeat” style of status reporting. Ex. B-7, 2:21-28; Ex. B-8, ¶0008-0009; Ex. B-9, ¶0001. The use of regular, periodic “heartbeat” status reports from a client to a server was very well known and a preferred manner for maintaining status information about a client at the server in a networked environment.

2. Claim 1

- a. Claim 1 preamble: “A method for tracking digital media presentations delivered from a first computer system to a user’s computer via a network comprising:”**

110. It is my opinion that Davis discloses this feature. As discussed previously, Davis discloses a technique for tracking the display of webpages, and the content embedded therein, on a Client computer. Davis, 11:34-13:17.

111. The display of content, such as audio or video, in one of these webpages would be a “digital media presentation.” This is true at least because audio and video are types of media, which are typically delivered in digital format in Web-based applications like in Davis. In addition, when the audio or video is played-back, this would be a form of presentation of the content. Davis explains that the audio and video content is delivered across a network, such as the Internet. Davis, 3:14-32, 5:4-13, 6:14-34, 11:34-12:5.

112. Davis also describes a “first computer system.” As explained previously in this declaration, Davis describe both a Server A and a Server B. Davis, 11:34-12:5. The Server A provides the webpage, while the Server B provides the timer Applet and stores the tracking information. Davis, 11:34-12:50.

113. While Davis does not explicitly describe Server A and Server B as being part of the same “system,” Davis’s description of Server A and Server B was

consistent with the way that a person of ordinary skill would have understood a “system” as of August 2008. In particular, a person of ordinary skill in the field would have understood a “system” in the Web context to often include more than one computer, as well as computers performing different but complimentary roles. For instance, a website operator’s “system” might include a web server for serving web resources, a database server for persisting website information, and any number of other servers for providing complimentary functionality. As a further example of the use of larger “systems” of servers, larger web sites would often be implemented in multiple web servers with a load-balancing server that directs client requests to the least utilized server. In this vein, Davis’s description of what amounts to a web server, Server A, and a tracking server, Server B, is wholly consistent with what would have been a person of ordinary skill’s notion of a “system” in the Web context in August 2008.

114. This “system” notion of Server A and Server B is further supported by Davis’s disclosure. For instance, Davis says that the functionality of Server A and the functionality of Server B could, at least in part, be combined together. Davis, 12:33-50, 17:63-18:7. Davis says that the tracking program could be provided on the same server as the webpage. Davis, 17:63-18:7. Davis says that the database of tracking information can be stored on Server B “or elsewhere.” Davis, 12:33-

50. A person of ordinary skill would have understood from these descriptions in Davis that the distinction between Server A and Server B need not exist, as long as the functionality was provided somewhere. And a person of ordinary skill would have found no technical challenge in moving a database, a tracking program, and two CGI scripts from one server to another.

115. In addition, Davis described the same sort of entities operating on Server A and Server B. Davis, 4:15-18, 11:24-33. Namely, Davis says that “network administrators” and “web site administrators,” both of which could be associated with the webpages hosted on Server A, would be consumers of the tracking information stored on Server B. This further supports the notion that Server A and Server B are in the same system under the ’609 Patent’s definition of “computer system” as being multiple devices “having a common operator or under common control.” As a result of these suggestions in Davis, in combination with the reasons described above, a person of ordinary skill in the field in August 2008 would have understood Server A and Server B to be part of a single “computer system.”

116. It is also true that, even without relying on Davis’s disclosure that the two servers can form one computer system, a person of ordinary skill would have nonetheless found it quite obvious to modify Davis’s system so that they were part

of the same “computer system.” Namely, for the reasons described above, a person of ordinary skill would already have understood that Server A and Server B were complimentary to one another, were operated on by similar entities, and could exchange functionality responsibilities with one another--all based on the disclosure of Davis itself.

117. Given this background, a person of ordinary skill would have understood that it would be beneficial to have a single entity operate and/or control both Server A and Server B. Namely, if one considers an entity that is a website administrator, that website administrator would already operate and/or control what is described as Server A in Davis. That website administrator, assuming he/she meets the requisite criteria of a person of ordinary skill in the field, would be motivated to implement the tracking functionality of Davis, due to the monetary benefits and increased information benefits explained in Davis. In so doing, the website administrator would then operate and/or control what is described as Server B in Davis. Being under common operation and/or control, the Server A and Server B would thereby be in the same “computer system.” This is just one example of why a person of ordinary skill would have found it obvious to provide Server A and Server B as a single computer system, and other reasons no doubt also exist.

b. Claim 1.a: “providing a corresponding web page to the user’s computer for each digital media presentation to be delivered using the first computer system;”

118. It is my opinion that Davis discloses this feature. As discussed above, the Server A in Davis’s system is the server that provides the webpage. Davis, 11:34-12:5. And as discussed above, the webpage provides a digital media presentation, such as with audio or video. Davis, 7:1-29, 11:34-12:5. And, as also discussed above, the Server A is part of the “first computer system” along with Server B.

119. Focusing on any one digital media presentation, i.e., any given audio or video resource, there is a corresponding webpage that is provided by Server A. Namely, the only digital media presentations contemplated by Davis’s disclosure are the ones that are embedded on some webpage. Davis, 7:1-29, 11:34-12:5. Audio or video that is not embedded in some webpage would be of no relevance to Davis’s system, as there would be no way for it to be presented to the user.

120. Hence, every resource of audio or video in Davis’s system is embedded in some webpage, and so it has some corresponding webpage.

c. Claim 1.b: “providing identifier data to the user’s computer using the first computer system;”

121. It is my opinion that Davis discloses this feature. Davis actually describes providing two separate types of identifier data to the Client from the Servers A and B.

122. First, Davis describes the Server A providing “embedded URLs” in the webpage to the Client. Davis, 11:34-12:5. These embedded URLs identify resources that the Client will include in the presentation of the webpage. Davis, 11:34-12:5. As discussed previously in this declaration, Davis describes these resources as images displayed in the webpage for Figure 4, but a person of ordinary skill would understand that these resources could also be audio or video content. Davis, 7:1-29, 16:63-17:10.

123. Second, Davis describes the Server B providing a “client ID” to the Client as part of the CGI Script 1. Davis, 11:34-12:5. As discussed previously in this declaration, the CGI Script 1 is used by the Client and Server B to basically register the interaction of the Client with the Server B. Davis, 11:34-12:5. Part of that registration is that the Server B checks whether the Client provided a client ID. Davis, 11:34-12:5. If it did not, then the Server B generates one and provides it to the Client in the response by the CGI Script 1. Davis, 11:34-12:5.

124. It is also my opinion that when streaming content was used in the webpage described in Davis, a person of ordinary skill in the field as of August 2008 would have been motivated to provide a “stream identifier” as described in Choi in the communication between Server A and the Client in Davis.

125. As introduced previously, Choi described that a “session identifier” and a “stream identifier” would be exchanged between the client and the server as part of the technique disclosed in Choi. Choi, ¶0044, Figure 3. The session identifier was used to identify the interaction between the client and the server, and the stream identifier was used to identify what stream of content the client was receiving. Choi, ¶0044, Figure 3.

126. The session identifier and stream identifier described in Choi are very similar to the client identifier and resource-identifying “embedded URLs” described in Davis. In essence, the session identifier described in Choi performs largely the same function as the client ID in Davis: they both identify the interaction between the client device and the server device. Choi, ¶0044, Figure 3; Davis, 11:34-12:5.

127. Likewise, the stream identifier in Choi and the embedded URLs in Davis are quite similar in function. The stream identifier in Choi identifies what stream of content the client is receiving. Choi, ¶0044. The embedded URLs in

device identify what resources the Client should be retrieving and presenting.

Davis, 11:34-12:5. Thus they both identify the actual content to be presented to the user.

128. Davis's use of the terminology "embedded URLs" with respect to Figure 4 is something of an artifact of the example Davis is describing. As discussed previously in this declaration, while Davis contemplates providing streaming media in the webpage provided to the client, the example given in the description of Figure 4 is just static images. Davis, 11:34-12:5. And when the resource being presented is simply an image, then an embedded URL would be sufficient to allow the Client to retrieve and present that content.

129. But when the webpage in Davis was used to provide streaming content, like audio or video, as described previously in this declaration, a person of ordinary skill in the field as of August 2008 would have recognized that some other identifier may be used to carry out the basic interactions described in Davis. Namely, if the content being presented in the webpage was a stream of content, then a stream identifier, such as described in Choi, would likely be provided. That stream identifier might itself be a URL, but it would also be a stream identifier given that it would be identifying a stream of content being displayed.

130. A person of ordinary skill in the field would also be motivated to have the Server A of Davis provide a stream identifier to the Client because, as described later in this declaration, a person of ordinary skill would recognize that there would be additional types of information that would be worth tracking and reporting to Server B when the content being presented is a stream of content. In order to associate that additional tracking information with the correct content, a person of ordinary skill would recognize that it would be useful to provide an identifier for that content, a stream identifier.

d. Claim 1.c: “providing an applet to the user’s computer for each digital media presentation to be delivered using the first computer system, wherein the applet is operative by the user’s computer as a timer;”

131. It is my opinion that Davis discloses this feature. Davis describes providing a “JAVA applet, the tracking program” from Server B to the Client. Davis, 12:13-50. The tracking program is then used as a timer to track how long the webpage is displayed. Davis, 12:13-50. The Client then reports that tracking information to Server B. Davis, 12:13-50. Thus, Davis describes this feature.

- e. **Claim 1.d: “receiving at least a portion of the identifier data from the user’s computer responsively to the timer applet each time a predetermined temporal period elapses using the first computer system; and”**

132. It is my opinion that Davis disclosed some parts of this feature, and that the rest would have been changes that a person of skill in the field would have been motivated to add to Davis’s system based on what Choi discloses.

133. The most relevant disclosure of Davis with regarding to Figure 4 for this feature is Davis’s explanation of the tracking program (the Applet) stopping and then the Client invoking CGI Script 2 on Server B. Davis, 12:5-50. In this portion, the user navigates away from the webpage, and the tracking program timer is stopped. Davis, 12:5-50. The Client invokes the CGI Script 2 on Server B, which is how the Client provides the tracking information about the duration the webpage was displayed to the Server B. Davis, 12:5-50. Besides the tracking information, the Client also includes an HTTP request header. Davis, 12:5-50. Earlier in the disclosure, Davis explained that the HTTP request header would include various identifying information for the Client, including the client ID. Davis, 11:59-12:5.

134. In sum, Davis discloses the first computer system (of which Server B is a part) receiving identifier information in the form of the client ID when the

when the timer Applet in the form of the tracking program determines that a period of time has ended, in the form of the time the webpage is displayed ending.

135. However, there is one part of this claim feature that Davis does not describe. Davis does not describe that the tracking program, the timer Applet, operates for a **predetermined** temporal period. But as described previously, a person of ordinary skill in the field as of August 2008 would have been motivated to modify Davis to use the periodic reporting approach from Choi. Choi, ¶¶0047, 0097. Choi in particular discloses that the reporting interval duration is predetermined during the “initial request” exchanged between the client and the server. Choi, ¶¶0097. Therefore, based on the explanation given previously in this declaration, a person of ordinary skill in the field would have been motivated to have the Client in Davis send tracking information to the Server B in Davis “each time a predetermined temporal period elapses.”

136. Based on my previous explanations of the ways in which a person of ordinary skill in the field as of 2008 would have been motivated to modify Davis, there are several other elements of tracking information that a person of ordinary skill would be motivated to include in the message from the Client to the Server B. This is particularly the case when the content presented in the webpage was streaming content, as discussed previously. With streaming content, there would

be additional elements of information that might be useful to include in Server B's tracking database.

137. First, a person of ordinary skill in the art would have been motivated to include the "stream identifier," as disclosed in Choi, in the message from the Client to Server B in Davis. As discussed previously in this declaration, a person of ordinary skill would have been motivated to send a stream identifier from Server A to the Client as a way to identify the streaming content that the Client would be presenting. With such a modification, a person of ordinary skill would also have been motivated to include that stream identifier in the tracking information from the Client to Server B. Davis already disclosed that multiple elements of identifier information would be included in that message. Davis, 11:59-12:5, 12:33-50. Such identifier information would be necessary for Server B to associate the tracking information with some client, some content, some webpage, etc. Thus, when a stream identifier was one of the relevant elements of identifier information for the tracking being performed at the Client, the Client would also include that stream identifier in the tracking information reports to Server B.

138. Second, when the content being presented in the webpage was streaming content, a person of ordinary skill would have recognized that streaming content has additional metrics about which usage can be tracked. In particular,

when Davis describes presenting static images in the webpage, the only metric Davis describes tracking is duration of display. Davis, 11:34-12:50. This makes sense, given that, at least when Davis was filed, there were not many more interactions with a static image that a client could be expected to be able to monitor. Duration of display was one of only a few possible metrics for a static image (e.g., along with number of visits to the webpage).

139. But when Davis describes displaying streaming content, Davis describes additional metrics that can be tracked. Davis, 16:63-17:10. Davis describes tracking the amount of data transferred and the amount time for which content is displayed. Davis, 16:63-17:10. Davis describes the client providing this additional tracking information back to the server. Davis, 16:63-17:10. Therefore, a person of ordinary skill would have recognized that additional metrics could be tracked when the content was being streamed, even within the bounds of Davis's own description.

140. A person of ordinary skill could then find in a reference like Choi a listing of relevant tracking parameters for streaming content. Just as a few examples, Choi describes having the client report the following tracking information to the server:

141. (1) “c-starttime”: “The timestamp (in seconds, no fractions when you played the file in normal mode.” Choi, Table C1.

142. (2) “c-endtime”: “The timestamp (in seconds, no fractions) when the Client finished playing the file in normal mode.” Choi, Table C1.

143. (3) “x-duration”: “How long you tried to receive the stream (in seconds).” Choi, Table C1.

144. (4) “Avgbandwidth”: “Average bandwidth (in bytes) at which the Client was connected to the Server.” Choi, Table C1.

145. (5) “c-bytes”: “Number of bytes received by the Client form the Server.” Choi, Table C1.

146. Choi describes a client sending this tracking information back to the server providing the content, but a person of ordinary skill would have recognized that it would have been valuable to also store such tracking information in a dedicated tracking information location, like the Server B of Davis. This is clear for a few reasons.

147. First, Davis itself says that information such as quantity of streaming data or duration of presenting streaming data is something that should be stored in a tracking server. Davis, 16:63-17:10. Davis explains that this information might be used to perform billing to the user of the client machine. Davis, 16:63-17:10.

148. Second, Davis explains that detailed tracking information about the time of viewing a webpage could be useful to website administrators to determine the popularity of particular webpages, as well as to set advertising rates for advertisements embedded in those webpages. Davis, 16:15-62. The same holds true for streaming content. Determining how long streaming content was presented, how long streaming content was transferred, and how much streaming content was transferred would allow website administrators and/or owners of that streaming content to determine the popularity of its specific streaming content. Further, with advertisements embedded in streaming content (e.g., a 30-second video advertisement presented during a video stream), the website administrators and/or owners of the streaming content could more effectively determine advertising rates for the streaming content (e.g., require higher rates from advertisers for displaying advertisements in highly popular video streams).

149. Third, Davis disclosed elsewhere that tracking the duration that a webpage is displayed would be useful to advertisers in order to “make informed decisions as to the effectiveness and value of particular Web pages and/or ad banner.” Davis, 11:13-33. The same holds true for streaming content, for the reasons just described for website administrators and content owners. Just as website administrators and content owners would find value in more detailed

tracking information so as to determine content popularity and inform how much to charge for advertising, advertisers would want the same information. Davis, 11:13-33. Advertisers are on the other end of the bargain as the website administrators and/or content owners, so they too would want to make an informed decision as to how much to pay to place advertisements in various streaming content (e.g., demand a discount for advertisements placed in streaming content that is not very popular).

150. For these reasons, it is my opinion that a person of ordinary skill in the field as of August 2008 would have been motivated to include the detailed tracking information parameters described in Choi, including those listed above, in the tracking information reports from the Client to Server B in Davis.

f. Claim 1.e: “storing data indicative of the received at least portion of the identifier data using the first computer system;”

151. It is my opinion that Davis discloses this feature. Davis discloses that the Server B stores the tracking information that it receives from the Client as part of the CGI Script 2 interaction. Davis, 12:33-50. The Server B stores the information in a database. Davis, 12:33-13:18. The information is retained in the database so as to be used by the website administrators, advertisers, etc. that Davis describes as the consumers of this tracking information. Davis, 12:51-13:18, 11:13-33.

152. Earlier in this declaration, I explained why a person of ordinary skill in the field as of August 2008 would have been motivated to have the Client send additional tracking information to the Server B when the content being presented was streaming content. As I explained, this additional tracking information would be valuable to website administrators, advertisers, content owners, etc. Thus it is necessary that the Server B store this additional tracking information when the Server B receives it from the Client. Otherwise, there would be no way for the website administrators, advertisers, content owners, etc. to ever actually use the tracking information. Therefore, it is my opinion that a person of ordinary skill in the field, when modifying Davis to send additional streaming-specific tracking information from the Client to the Server B, would also modify Server B to store that additional tracking information.

- g. Claim 1.f: “wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from the first computer system directly to the user’s computer independent of the first computer system;”**

153. It is my opinion that Davis discloses this feature. As explained previously, Davis describes the process of Figure 4 as involving the Server A providing a webpage with embedded URLs referencing resources, and then the Client fetches the resources at the URLs and displays the webpage. Davis, 11:33-12:5. As also explained previously, while Davis only describes Figure 4 with the

example of static images being the resources referenced in the webpage, Davis makes clear elsewhere that the resources can be audio and video. Davis, 7:1-28. And as explained previously, a person of ordinary skill in the field as of August 2008 would have been motivated to provide this audio/video in streaming format, either based on Davis's own disclosure or as a modification based on the disclosure of Choi. Davis, 16:63-17:10; Choi, ¶¶0002-0012.

154. Therefore, the only remaining aspect of this feature not discussed previously in this declaration is whether the content streamed to the Client in Davis would be streamed "directly" to the client from a "second computer system" that is "distinct" from the first computer system (including Servers A and B).

155. First, in Davis's system, the resource fetched by the Client comes "directly" from whatever server is hosting it. Davis discloses that the Client fetches resources using the embedded URLs. Davis, 11:34-51. What this means is that the Client accesses the embedded URL, which would point to whatever server is hosting the resource. When the embedded URL (or stream identifier) pointed to a streaming resource, the process would be the same. Hence, Davis does disclose that the Client streams content "directly" from whatever server is hosting it.

156. Second, Davis also discloses that the streaming resource can come from a "second computer system" that is "distinct" from Servers A and B. For

Figure 4, Davis does describe the images as potentially being stored on Server A, which would not be a second computer system distinct from the first computer system. Davis, 11:33-52. But Davis also discloses that the images can be stored on “any HTTP server on the Internet.” Davis, 11:40-45. In describing the same feature with respect to Figure 3, Davis discloses that the embedded URLs can point to “resources located on any server.” Davis, 9:16-33. When the referenced resource is stored on “any server” or “any HTTP server on the Internet,” that resource clearly can be stored on some server that is not part of the computer system of which Servers A and B are a part. To be otherwise would require that Server A, Server B, and every other computer on the Internet are in the same “first computer system,” which does not appear to be what the claim is reciting.

157. Hence, Davis disclosed this feature, including the part whereby the streaming content comes from a second computer system directly to the Client.

- h. Claim 1.g: “wherein the stored data is indicative of an amount of time the digital media presentation data is streamed from the second computer system to the user’s computer; and”**

158. It is my opinion that the system described in Davis, when modified as explained previously in this declaration, would include this feature. As explained previously, while Davis only describes Figure 4 with the example of static images being the resources referenced in the webpage, Davis makes clear elsewhere that

the resources can be audio and video. Davis, 7:1-28. As also explained previously, a person of ordinary skill in the field as of August 2008 would have been motivated to provide this audio/video in streaming format, either based on Davis's own disclosure or as a modification based on the disclosure of Choi. Davis, 16:63-17:10; Choi, ¶0002-0012. As also explained previously, a person of ordinary skill in the field would have been motivated to include additional tracking information parameters in the tracking reports from the Client to the Server B when the webpage was displaying streaming content. Choi, Table C1. And as explained previously, the Server B would store those additional tracking information parameters. Davis, 12:33-13:18. Because these additional tracking information parameters include parameters that indicate how long streaming content was streamed from the third-party server (that is, the second computer system as explained previously), the modifications to Davis's system described previously would result in the system including this feature.

159. As explained previously, the streaming-specific tracking information that a person of ordinary skill would have been motivated to add to the tracking reports from the Client to the Server B in Davis are described in Table C1 of Choi, and include the following, among others:

160. (1) “c-starttime”: “The timestamp (in seconds, no fractions when you played the file in normal mode.” Choi, Table C1.

161. (2) “c-endtime”: “The timestamp (in seconds, no fractions) when the Client finished playing the file in normal mode.” Choi, Table C1.

162. (3) “x-duration”: “How long you tried to receive the stream (in seconds).” Choi, Table C1.

163. (4) “Avgbandwidth”: “Average bandwidth (in bytes) at which the Client was connected to the Server.” Choi, Table C1.

164. (5) “c-bytes”: “Number of bytes received by the Client form the Server.” Choi, Table C1.

165. As I explained previously, it is my opinion that “an amount of time” data was “streamed” should be interpreted to mean “an amount of time” the data was “transferred via a technique such that the data can be processed as a substantially steady or continuous sequence.” But as also explained previously, I will analyze this feature under the alternative interpretation that “an amount of time” the data was “streamed” means “an amount of time” the data was “presented” on the user’s computer.

166. If “streamed” means “transferred via a technique such that the data can be processed as a substantially steady or continuous sequence,” then the

modified system of Davis using the parameters from Table C1 in Choi would actually include this feature in two distinct ways.

167. First, the parameter “x-duration” is indicative of the amount of time the streaming content was “streamed” to the client device. Choi, Table C1. Choi describes the “x-duration” parameter as capturing “How long you tried to receive the stream (in seconds).” Choi, Table C1. A person of ordinary skill in the field would understand that this means the amount of time the client device received the stream of content. The word “tried” used here is significant to Choi’s disclosure, because Choi deals with situations where there are network disruptions. Choi, ¶0002-0014. Hence, the client device does not always receive the stream of content. In any case, if there was a network disruption, the client would not be able to transmit a tracking report to the server.

168. But at least under normal operating conditions where there is not a network disruption, the “x-duration” value transmitted from the client to the server would indicate how long the client received the stream of content. And in such conditions where there is not a network disruption, the amount of time that the client received the stream is essentially the same as the amount of time the stream was transferred over the network from the server to the client.

169. To the extent a person of ordinary skill in the field were looking for further information on the meaning of the x-duration parameter, a guide such as the July 2008 Windows Media Log Data Structure specification would provide additional information. Ex. B-6 (“July 2008 MS-WMLOG Specification”). The July 2008 MS-WMLOG Specification explains that it defines “a syntax for logging messages.” July 2008 MS-WMLOG Specification, p. 5. “The logging messages specify information about how a client received multimedia content from a streaming server. For example, logging messages can specify how many packets were received and how long it took for the client to receive the content.” July 2008 MS-WMLOG Specification, p. 5. A MS-WMLOG specification, whether the July 2008 version or an earlier one, appears to be the source of the parameters listed in Table C1 of Choi.

170. The July 2008 MS-WMLOG Specification describes the x-duration parameter similar to Choi. It states that the parameter “MUST specify the time it took to receive the content, in seconds.” July 2008 MS-WMLOG Specification, p. 27. This would confirm for a person of ordinary skill in the art that storing the x-duration parameter in the tracking information database of Server B would result in stored data indicative of the amount of time streaming content was streamed to the Client. The Jun 2008 version of the MS-WMLOG specification would also have

confirmed this understanding by a person of ordinary skill in the field. Ex. B-5 (“June 2008 MS-WMLOG Specification”).

171. Second, the combination of parameters “Avgbandwidth” and “c-bytes” are indicative of the amount of time the streaming content was “streamed” to the client device. Choi, Table C1. Choi describes Avgbandwidth as “Average bandwidth (in bytes) at which the Client was connected to the Server.” Choi, Table C1. Choi describes c-bytes as “Number of bytes received by the Client from the Server.” Choi, Table C1. A person of ordinary skill in the field would recognize that the amount of data transferred divided by the bandwidth would equal the amount of time that it took to transfer the data. This is inherent in the meaning of bandwidth. Further, even with a statistic of the bandwidth, i.e., the average, this quotient would still be indicative of the amount of time that data transfer took place.

172. The person of ordinary skill in the field as of August 2008 would have this understanding confirmed by the MS-WMLOG specifications. For instance, the July 2008 MS-WMLOG Specification described Avgbandwidth as “the average bandwidth, in bits per second, at which the client received content from the server.” July 2008 MS-WMLOG Specification, p. 9. This was measured “by the client from the start of the current session.” July 2008 MS-WMLOG

Specification, p. 9. “This is only applicable during periods in which the server is streaming the content.” July 2008 MS-WMLOG Specification, p. 9. The June 2008 MS-WMLOG Specification has similar confirmatory descriptions. June 2008 MS-WMLOG Specification, p. 9.

173. Both specifications have similar confirmatory descriptions for the c-bytes parameter. July 2008 MS-WMLOG Specification, p. 18; June 2008 MS-WMLOG Specification, p. 18.

174. Therefore, either in the form of the x-duration value stored by Server B, or in the form of the stored Avgbandwidth and c-bytes values stored by Server B, the data stored by Server B would indicate the amount of time the digital media presentation data was streamed, i.e., transferred, from the server hosting the streaming content to the Client.

175. If, on the other hand, “streamed” is interpreted to mean “presented,” then the person of ordinary skill in the field as of August 2008 would recognize that the combination of the “c-starttime” parameter and “c-endtime” parameter described in Choi would indicate the amount of time the streaming content was “streamed,” i.e., presented. Choi, Table C1. Choi describes c-starttime as “The timestamp ... when you played the file in normal mode.” Choi, Table C1. Choi describes c-endtime as “The timestamp ... when the Client finished playing the file

in normal mode.” Choi, Table C1. A person of ordinary skill in the field would have recognized that the difference between c-endtime and c-starttime would indicate how long the file was played. Thus this difference would indicate how the amount of time the digital media presentation data was streamed, i.e., presented.

176. I also note that if the amount of time that content is “steamed” means how long it is presented, then Davis itself actually teaches storing this tracking information even before any modifications based on Choi. Davis, 16:63-17:10. As explained at length previously in this declaration, Davis disclosed that when the content is streaming media, the tracking program should track “the time the information is displayed.” Davis, 16:63-17:10. Davis discloses that this information should be tracked because sometimes a user is charged based on “the amount of information displayed, either according to bit size or time.” Davis, 16:63-17:10. For this additional reason, when Davis’s system described with respect to Figure 4 was included streaming content in the webpage, a person of ordinary skill would recognize that the tracking information database stored by Server B would include stored data that indicates the amount of time the streaming content was “streamed,” i.e., presented.

i. Claim 1.h: “wherein each stored data is together indicative of a cumulative time the corresponding web page was displayed by the user's computer.”

177. It is my opinion that the system described in Davis, when modified as explained previously in this declaration, would include this feature. As explained previously, Davis disclosed using the tracking program to track how long a webpage was displayed on the Client, and that tracking information was transmitted to the Server B, which stored it. Davis, 12:5-50.

178. Further, to the extent that this feature requires that the Server B stores multiple tracking information entries for a single webpage visit, a person of ordinary skill in the field would have recognized that this would result when Davis was modified based on Choi's disclosure, as explained previously in this declaration, to use a predetermined timing interval, as opposed to the event-based reporting approach disclosed in Davis. Namely, any time that the user of the Client remained on the webpage for a longer period of time than the predetermined reporting interval, Choi, ¶0097, the Client would send at least two tracking reports to the Server B. Hence, the two or more tracking reports stored by the Server B would together indicate how long the webpage was displayed on the client.

179. A person of ordinary skill in the field would have recognized that the storage of multiple tracking records would actually be expected, even desirable, when Davis was modified to use a predetermined reporting interval. As explained

previously in this declaration, one aspect of the predetermined reporting interval that would have motivated a person of ordinary skill in the field to use it was the fact that more frequent tracking reports could be sent to the Server B. The benefit, as explained, was that there would be more frequent updates on information and less risk of delaying or even losing tracking information. Hence, a person of ordinary skill in the field would have been motivated to use a relatively short predetermined reporting interval, and thus increase the likelihood that two or more tracking information records would be stored by Server B for a single webpage visit.

3. Claim 2: “The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.”

180. It is my opinion that when the system described in Davis was modified as explained previously in this declaration by as taught by the disclosure of Choi, a person of ordinary skill in the art would have been motivated to include this feature in Davis’s system. As explained previously in this declaration, a person of skill in the art would have been motivated to modify Davis’s system so that the Client would send tracking reports to the Server B according to a predetermined timing interval.

181. Davis already disclosed that one way to store tracking information was to increment a counter each time a tracking report was received by the server.

Davis, 3:42-44. Davis disclosed that, in order to track the number of times that some particular content, such as an advertisement, had been displayed, the server could increment a stored counter. Davis, 3:42-53. The same approach could be used to track the number of times an advertisement was clicked on. Davis, 3:42-53. Thus, a person of ordinary skill would have recognized that it would be possible to increment a stored counter in Server B of Davis in order to store some types of tracking information.

182. When Davis's system was modified as explained previously in this declaration, a person of ordinary skill would have been motivated to use an incrementing approach for storing various types of tracking information. For example, instead of storing multiple tracking records for a single webpage visit, each indicating the same predetermined timing interval, the Server B could simply increment a value for "number of intervals" as well as the length of the predetermined timing interval. For instance, instead of storing 10 tracking records, each indicating "15 seconds," the Server B could store one tracking record, which would indicate "number of intervals" as "10" and "length of interval" as "15 seconds."

183. A person of ordinary skill would have recognized that the Server B may still need to store a second tracking record, for example, when the last

tracking report for the webpage visit indicated a length of the display of the webpage that was less than the full reporting interval. But a person of ordinary skill would have recognized that this would allow a reduction of the number of tracking records from 11 (continuing the previous example) down to two.

184. A person of ordinary skill would have been motivated to use an incrementing approach as disclosed in Davis for a few reasons.

185. First, Davis already disclosed that this approach was possible, and a person of ordinary skill would have been likely to choose a familiar approach recommended by Davis itself.

186. Second, use of a counter is something that any undergraduate student in computer science or a like field would be very familiar with. Thus, a person of ordinary skill, who would have at least that level of training, would be motivated to use known, simple solution to the problem.

187. Third, a person of ordinary of ordinary skill in the field would have recognized that the incrementing approach would allow a significant reduction in the size of the database stored by Server B. In the example given just above, the reduction was on the order of a five-fold decrease in the number of records stored. This reduction in storage space would provide some marginal benefit to being able

to store more tracking information in the Server B using the same server specifications.

188. Fourth, a person of ordinary skill in the field would have recognized that using the incrementing approach would make it easier and quicker to process many of the tracking reports that the Server B received from the Client. In particular, when the Server B received a tracking report from the Client as part of CGI Script 2 being invoked by the Client, the Server B could execute a simple update command on the database to increase the “number of intervals” value. If the tracking database used SQL, for example, a single UPDATE statement could be used to store the new tracking information. On the other hand, storing each tracking report as a separate tracking record would require forming and then inserting a new record into the tracking database each time a tracking report was received. A person of ordinary skill would have recognized that this would have required more logic to perform, would introduce more possibilities for error, and might have consumed more resources on the database.

189. Fifth, a person of ordinary skill would have recognized that storing fewer tracking information records for each webpage visit would allow simpler and quicker aggregation of tracking information when the database was later used by its consumers (e.g., website administrators, advertisers). With the incrementing

approach, the program that aggregated tracking information for consumption by the website administrators and advertisers would need to process fewer tracking records from the database, which would make the aggregation process simpler and shorter.

4. Claim 3: “The method of claim 2, wherein the received data is indicative of a temporal cycle passing.”

190. It is my opinion that the system described in Davis, when modified as explained previously in this declaration, would include this feature. As explained previously, a person of ordinary skill in the field in August 2008 would have been motivated to modify the system of Davis to use a predetermined reporting interval as disclosed in Choi. Choi, ¶¶0047, 0097. As explained previously, the Server B of Davis would have received a tracking report at the end of each predetermined reporting interval when the Client invoked CGI Script 2. As such, when the Server B received the tracking information due to invocation of CGI Script 2, that receipt would indicate that a temporal cycle, i.e., the predetermined reporting interval, had passed.

C. Ground II: Claims 1-3 were obvious in view of Siler and Davis

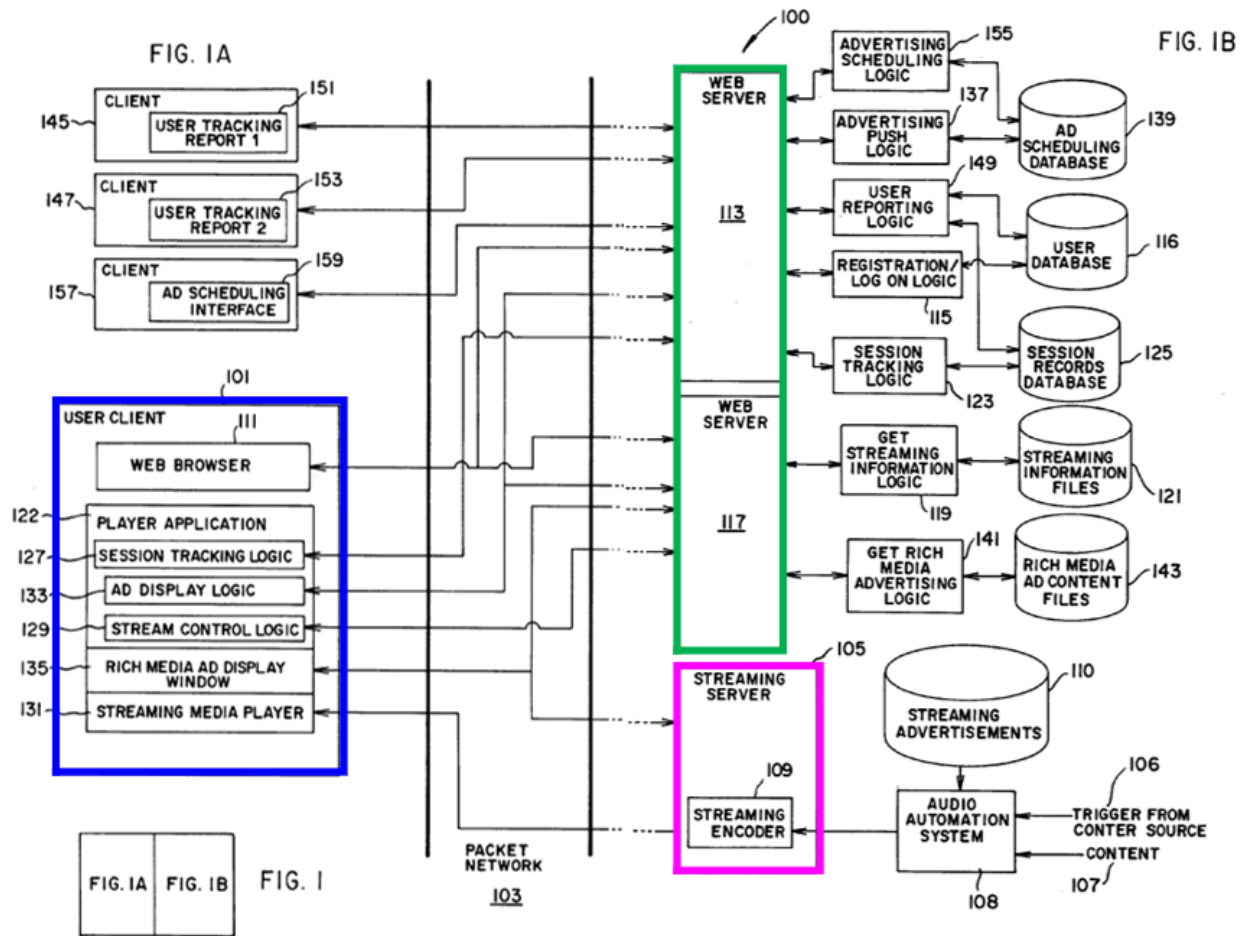
1. The Siler-Davis Combination

a. Siler

191. The Siler reference disclosed a system that handles streaming media over a packet switched network. Siler, Abstract. The system tracks “which users are receiving a particular media stream and how long each of the users receives” it. Siler, Abstract. Advertisements are added to the stream in real-time based on a trigger signal. Siler, Abstract.

192. Siler disclosed a system for streaming media from a server to a client that would track “which users are receiving a particular media stream and how long each of the users receives” a particular media stream. Siler at Abstract.

193. Siler described its system using Figures 1A and 1B, which I understand are essentially two halves of a composite Figure 1. The composite Figure 1 is shown below, with Figures 1A and 1B combined side-by-side.



194. Siler's system includes three major components, at least with respect to the subject matter of the '609 Patent's claims. A user client 101 is what a user uses to receive and view streaming content. Siler, ¶¶0017, 0023-0029. A streaming server 105 streams the content, including advertisements, to the user client 101. Siler, ¶¶0017-0022, 0027. The combination of web server 113 and web server 117 handle various interactions with the user client 101, including storing tracking information gathered by the user client 101. Siler, ¶¶0023-0030.

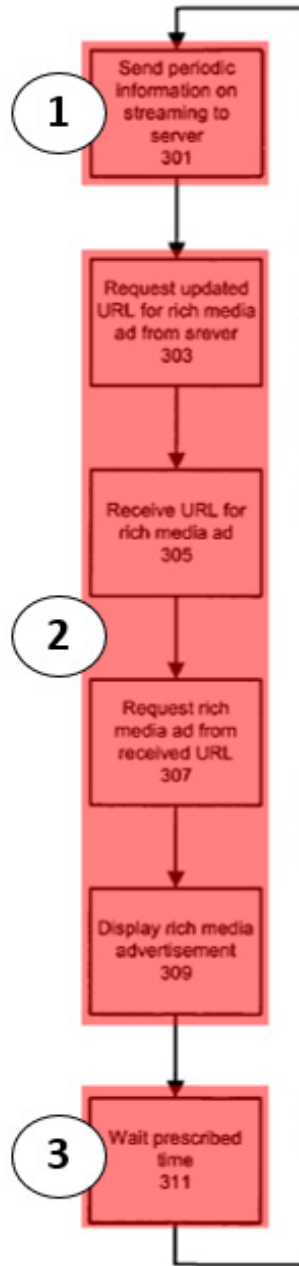
195. Siler describes how the system initially sets up streaming media for presentation at the user client 101.

196. This starts with the user client 101 displaying a webpage that has links to various streaming content that is available. Siler, ¶0023. The user clicks one of these options. Siler, ¶0023. This causes an interaction with web server 113, which registers the user if the user is not already registered. Siler, ¶0023-0024.

197. The user client 101 then sends a request to the web server 117, requesting to set up streaming of content associated with the selected link. Siler, ¶0025. The web server 117 retrieves a file with information on the stream, and sends to file to the user client 101. Siler, ¶0025. The file includes a URL for the stream. Siler, ¶0025

198. The user client 101 then sends a request to start the stream using the URL. Siler, ¶0027. The URL points to the streaming server 105. Siler, ¶0027. The streaming server 105 transmits the stream of content to the user client 101, along with a stream identifier and/or session identifier. Siler, ¶0027. The user client 101 presents the stream of content for playback. Siler, ¶0027.

199. Siler discloses that the system performed usage tracking with respect to the streaming content. Siler discloses this this technique with respect to Figure 1, as well Figure 3, which is shown below with annotations.



200. Siler discloses that the process of Figure 3 is performed by the user client 101. Siler, ¶0028. At step 301, the user client 101 sends information to the web server 113. Siler, ¶0028. It does so “automatically” and “on a periodic basis.” Siler, ¶0028. The information sent to the web server 113 includes the stream

identifier, the session identifier, and a user identifier. Siler, ¶0028. At steps 303, 305, 307, and 309, the user requests and receives a URL for a rich media advertisement, receives the rich media advertisement, and then displays it. Siler, ¶0028. At step 311, user client 101 “waits for a prescribed time before repeating the process.” Siler, ¶0028. After waiting the prescribed time, the user client would then repeat the process starting at step 301.

201. Siler disclosed actions performed by the web server 113 as part of the process of gather usage tracking information, for which Siler includes Figure 4. First, it is the web server 113 that receives the periodic tracking information messages from the user client 101 as part of the process just described above. Siler, ¶0028-0029. The web server 113 waits to receive updated information from the user client 101. Siler, ¶0029. When the web server 113 receives the updated information from the user client 101, it “passes the information to session tracking logic 123, for updating the session record for the particular user in step 403.” Siler, ¶0029. Session tracking logic 123 is shown in Figure 1, and it is shown with access to session records database 125.

202. Though Siler does not specify in paragraph 0029 that the session record that is updated by the web server 113 is one that is stored in session records database 125, that is where a person of ordinary skill in the art would have

understood the record to be stored. That is true because the session records database 125 is the only data store shown connected to session tracking logic 123. Siler, Figure 1. And that is true because Siler says elsewhere that “session tracking logic 123 creates a record in session records database 125 to track the user’s session with the selected media stream.” Siler, ¶0026. So, a person of ordinary skill would have understood that when the user client 101 sends tracking information to the web server 113 according to Figure 3, the web server 113 causes the tracking information to be stored in session records database 125.

203. In addition to the above, web server 113 performs the additional steps of Figure 4 in order to deliver the URL for the rich media advertisement to the user client 101, as referenced in my previous explanation of Figure 3. Siler, ¶0029.

204. Siler also discloses client computers 145 and 147, shown in Figure 1, as computers that can view the tracking information stored by the web server 113. Siler, ¶0030. The tracking information can be viewed in real time. Siler, ¶0030. The tracking information includes information about “how many people are actually listening to the content, as well as when they listened and how long they listened.” Siler, ¶0030.

b. Davis

205. I have provided an explanation of Davis's disclosure previously in this declaration, and I refer back to that explanation for the remainder of this declaration.

c. Motivation to Combine

206. Based on review of Siler and Davis, and based on the background of the field in August 2008, it is my opinion that a POSA would have been motivated to combine several features described in Davis into the system described in Siler.

207. One significant reason that a person of ordinary skill would have looked to the disclosure of Davis to modify Siler's system is because of a change that Siler's own disclosure suggests.

208. In Siler's primary description of the user client 101, Siler describes a player application 122 that displays the streaming content and performs activities such as the usage tracking explained previously. Siler, ¶0025, 0027-0028. Siler discloses that the player application 122 could use a streaming media client 131 "such as Windows Media Player" to "actually controls the streaming and processes and decodes the stream ... for playback on client computer 101." Siler, ¶0027. In other words, Siler discloses that, while the web server 113 can provide a webpage from which a user selects a stream to view, Siler, ¶0023, the actual streaming of content can be performed in a separate software application. Siler, ¶0027.

209. But Siler also suggests that a variation on this arrangement of the user client 101 can be used. Siler, ¶0032. Namely, the “Player application 122 may, alternatively, be implemented as a web page with active components, with the rich media advertisements displayed in a frame.” Siler, ¶0032. In other words, Siler suggests that the application that controls the streaming and usage tracking on the client 101 could be implemented as a webpage in a browser. Siler, ¶0032.

210. A person of ordinary skill in the field as of August 2008 would have been motivated by this statement to modify Siler’s system at least in the way Siler suggests. Namely, a person of ordinary skill would be motivated to implement at least the session tracking logic 127 and stream control logic 129 as “executable content” in the web browser 111. As mentioned already, adding functionality to web pages by way of “executable content,” either in the form of Applets or scripts, was totally conventional at the time and a common way to add client-side functionality. Siler, Figure 1, ¶0027. Further, Siler’s primary description of user client 101 has a streaming media player 131 that may be “embedded in” the player application 122. Siler, ¶0027. So, a person of ordinary skill would also have recognized that when implementing the functionality of the player application 122 to be embedded within a webpage, per Siler’s suggestion, it would also be advantageous or even necessary to implement the streaming media player 131 in a

webpage. Siler, ¶0027, 0032. Thus, based on Siler's own disclosure, a person of ordinary skill in the field would have been motivated to implement the usage tracking functionality as well as the presentation of streaming content in a webpage to be viewed in the web browser 111 on the user client 101. One might implement each of them as a separate Applet, or one could even create a single Applet combining both of these functionalities.

211. Once a person of ordinary skill in the field was motivated to implement the player application 122 as a webpage, a person of ordinary skill would have also been motivated to make other changes that would be benefited by such a transition to the webpage implementation.

212. First, while Siler does not disclose that the timer logic of Figure 3 of the user client 101 is implemented as an Applet downloaded from the web servers 113 and 117, a person of ordinary skill in the field would have recognized that a conventional way of adding functionality to web pages was by way of "executable content" using either Applets or scripts. A person of ordinary skill in the art would also have recognized that this would be a beneficial change to make when the player application 122 was implemented in a webpage. This is true for several reasons.

213. One reason that a person of ordinary skill would have been motivated to implement the timer logic from Figure 3 as an Applet downloaded from the web servers 113 and 117 is because this approach was a well known design, as evidenced by Davis. Davis, 9:16-45, 10:11-57, 12:13-50. A person of ordinary skill would have been familiar with implementing logic in an Applet, and so a person of ordinary skill would have been motivated to implement the tracking logic timer using that familiar approach.

214. Another reason that a person of ordinary skill would have been motivated to implement the timer logic from Figure 3 as an Applet downloaded from the web servers 113 and 117 is because the person of ordinary skill would have recognized that the program in which it was previously provided, the player application 122, would no longer be pre-existing on the user client 111. Siler, ¶0032. Once the player application 122 is implemented as a webpage, it would be necessary to find a new “home” for the logic implementing the timer. A person of ordinary skill would have recognized that the most logical place to locate this timer logic would be still with the programming of the player application 122, which is now implemented in a webpage. A person of ordinary skill would have recognized that an Applet would be one way to implement the timer logic in relation to the

webpage performing the functionality of the player application 122, e.g., based on the disclosure of Davis. Davis, 9:16-45, 10:11-57, 12:13-50.

215. A person of ordinary skill in the field as of August 2008 would have recognized other benefits of using an Applet to implement the timing logic of Figure 3. For instance, an Applet implementation would have allowed Siler's system to require less software to be pre-installed on the user client 101. This would have made for a much more "seamless" user experience – installing and updating software is typically a disruptive activity to most users, and having to do this just at the time that you would rather want to watch a movie is annoying for many users. Also, a person of ordinary skill would have understood that an Applet implementation would have allowed Siler's system to implement the timer logic of Figure 3 in a platform-independent way. The timer logic could be downloaded and operated on the user client 101, regardless of what operating system and browser it used. Further, by implementing the timer functionality as an Applet served from the server, the system could keep evolving and updating the implementation of the timer functionality in a transparent manner while assuring that each user client 101 would always get the updated version each time the user client 101 loaded a webpage.

216. Second, another change that a person of ordinary skill would have been motivated to make to Siler's system when implementing the player application 122 as a webpage would be to increase the types of usage tracking performed in the system. Siler already disclosed tracking how many users accessed streaming content, when users accessed streaming content, and how long users accessed streaming content. Siler, ¶0030. Siler disclosed that this information would be tracked for the benefit of content providers and advertisers. Siler, ¶0030.

217. With the implementation of the player application 122 as a webpage, Siler, ¶0032, a person of ordinary skill would have also been motivated to track the duration that the webpage was displayed, because Davis disclosed that. Davis, 11:13-33, 12:13-13:17. A person of ordinary skill would have recognized that, with the introduction of a webpage to perform the functions of player application 122, there would be a new metric for which usage tracking could be informed. A person of ordinary skill would have recognized from Davis's disclosure that tracking the duration that this webpage was displayed would be beneficial to the very same users of the tracking information that Siler was already maintaining. Siler, ¶0030; Davis, 11:13-33, 12:51-13:18.

218. A person of ordinary skill would have found tracking the duration that the webpage was displayed particularly useful in the case where that webpage included an additional type of advertisement, such as a banner advertisement. Davis, 3:14-67, 11:13-33. A person of ordinary skill in the field would have recognized that Siler's system was designed to maximize advertisement presentation and monitoring. Siler, ¶¶0002-0006, 0017-0020, 0028-0029. Siler's system included advertisements spliced into the streaming media. Siler, ¶¶0017-0020. Siler's system included rich media advertisements, which could be displayed as pop-up images. Siler, ¶¶0028-0029. A person of ordinary skill would have recognized that, when the player application 122 was implemented as a webpage, additional advertisements could be displayed, such as banner advertisements in the webpage. Davis, 3:14-67, 11:13-33. And a person of ordinary skill would have been therefore motivated to track the display of those webpage advertisements using Davis's technique due to the benefits to advertisements and website administrators as disclosed in Davis. Davis, 3:14-67, 9:16-45, 11:13-33, 11:34-13:17.

2. Claim 1

- a. Claim 1 preamble: “A method for tracking digital media presentations delivered from a first computer system to a user’s computer via a network comprising:”**

219. It is my opinion that Siler discloses this feature. Siler describes tracking the streaming content delivered from the streaming server 105 to the user client 101. Siler, ¶0028-0030. A person of ordinary skill would have recognized that the streaming content was a “digital media presentation.” For instance, Siler describes the streaming content as “audio, and/or audio/video signals” that are transmitted over a packet network as a “data stream.” Siler, ¶0016-0017.

220. Siler also describes a “first computer system.” As explained previously in this declaration, Siler describes both a web server 113 and web server 117. Siler, Figure 1, ¶0024-0032. While Siler does not explicitly describe web server 113 and web server 117 as being part of the same “system,” Siler’s description of web servers 113 and 117 was consistent with the way a person of ordinary skill would have understood a “system” as of August 2008.

221. For instance, Siler described that “Web servers 113 and 117 do not necessarily correspond to physical machines. Rather, they represent different instances of a web server, which may or may not be running on the same physical

hardware.” Siler, ¶0032. Hence, Siler explicitly disclosed that web servers 113 and 117 could actually be implemented on the same computer device.

222. In addition, Siler described web servers 113 and 117 interacting with one another to perform the various registration, tracking, advertising, and other functions performed by them in Siler’s system. Siler, ¶0025, 0029. At least because of their functional interactions and the fact that they could be provided together on a single piece of hardware, a person of ordinary skill would have understood that web servers 113 and 117 could be provided as a single “system.” For instance, a single entity could operate and/or control the two web servers 113 and 117 due to their functional interrelation and ability to be deployed together.

b. Claim 1.a: “providing a corresponding web page to the user’s computer for each digital media presentation to be delivered using the first computer system;”

223. It is my opinion that Siler discloses this feature. Siler disclosed that the user client 111 receives a webpage with a list of streaming resources. Siler, ¶0023. Siler does not explicitly state where that webpage is provided from, but Siler explains that clicking on one of those links causes information to be sent to web server 113. Siler, ¶0023. A person of ordinary skill in the field would have recognized, based on this disclosure, that the web server 113 could have been provided by the web server 113. This is supported by the fact that a person of

ordinary skill in the field as of August 2008 would have understood that serving webpages was one of the functions typically performed by a “web server” as of August 2008.

224. As explained previously, Siler disclosed that the player application 122 could be implemented as a webpage. Siler, ¶0032. Siler does not explicitly state where such a webpage would be hosted. But based on the disclosure of Siler with respect to web server 113 serving other webpages to the user client 101, a person of ordinary skill in the field would have recognized that the web server 113 could also have served the webpage that implemented the player application 122. This is again supported by the fact that a person of ordinary skill would have recognized that a “web server” would typically have served a webpage as of August 2008. More so, a person of ordinary skill would have recognized that a “web server” would more typically have provided the functionality of serving a webpage than would have a “streaming server as of August 2008.

225. A person of ordinary skill in the field would have understood that the webpage implementing the functionality of the player application 122 and displaying the streaming content would be a webpage “corresponding” to that streaming content. First, when the player application 122 was implemented using a webpage, a person of ordinary skill would have recognized that each stream of

content provided by streaming server 105 would have to be presented in some webpage to be viewable by the user of the user client 101. As such, every stream of content would have a corresponding webpage. Second, Siler already disclosed that each stream of content could be identified by a different URL. Siler, ¶0025-0027. Hence, a person of ordinary skill in the field would have considered implementing a separate viewer webpage for each stream of content, in a way analogous to the separate URL for each stream as disclosed by Siler. Siler, ¶0025-0027.

c. Claim 1.b: “providing identifier data to the user’s computer using the first computer system;”

226. It is my opinion that Siler discloses this feature. Siler disclosed that the session tracking logic 123 provides a session identifier to the user client 101. Siler, ¶0026. A person of ordinary skill in the field would understand from the disclosure of Siler that the session tracking logic 123 would be part of the web server 113. Siler, Figure 1, ¶0026, 0029, 0032. Alternatively, a person of ordinary skill would have understood that Siler’s reference to the session tracking logic 123 providing a session identifier to the user client 101 would actually have entailed the session tracking logic 123 providing the session identifier to the web server 113, which would then provide it to the user client 101. Siler, Figure 1, ¶0026, 0029, 0032.

227. A person of ordinary skill in the field would have understood that the session identifier from Siler was “identifier data.” This is true at least because Siler states that the session identifier “uniquely identifies the session.” Siler, ¶0026. Siler also states that the “session identifier for a streaming session can be used in place of the stream ID to identify the stream that the user is then currently receiving.” Siler, ¶0027.

d. Claim 1.c: “providing an applet to the user’s computer for each digital media presentation to be delivered using the first computer system, wherein the applet is operative by the user’s computer as a timer;”

228. It is my opinion that Siler’s system, when modified as explained previously in this declaration based on the teaches of Siler and Davis, would have included this feature. As explained previously, a person of ordinary skill would have been motivated based on the teachings of Siler itself to implement the player application 122 as a webpage. Siler, ¶0032. And as explained previously, a person of ordinary skill, when implementing the player application 122 as a webpage, would have been motivated to implement the timer functionality of Figure 3 as an Applet downloaded from the web server 113 with the webpage. Siler, Figure 3, ¶0028. Thus, Siler’s system when modified as already explained in this declaration would include this feature.

229. In particular, the step 311 disclosed by Siler would be implemented by programming that would be operative as a timer. Siler describes step 311 as the user client 101 “waits for a prescribed time before repeating the process” of Figure 3. Siler, ¶0028. A person of ordinary skill in the field would have recognized that timer functionality would have been used to implement this step of waiting for a period of time.

- e. **Claim 1.d: “receiving at least a portion of the identifier data from the user’s computer responsively to the timer applet each time a predetermined temporal period elapses using the first computer system; and”**

230. It is my opinion that Siler’s system, when modified as explained previously in this declaration based on the teaches of Siler and Davis, would have included this feature. As explained previously, a person of ordinary skill, when implementing the player application 122 as a webpage, would have been motivated to implement the timer functionality of Figure 3 as an Applet downloaded from the web server 113 with the webpage. Siler, Figure 3, ¶0028. Siler disclosed that the timer functionality causes the user client 101 to “wait[] for a prescribed time.” Siler, ¶0028. A person of ordinary skill would have understood that a “prescribed” time was a type of “predetermined” time.

231. Siler also disclosed that the user client 101 transmitted, and the web server 113 received, tracking information each time the prescribed time elapses.

Siler, Figures 3- 4, ¶0028-0029. Namely, Siler discloses that, after the prescribed time elapses, the user client “sends periodic information on streaming to server.” Siler, Figure 3, ¶0028. At the same time, the web server 113 checks whether it has “Received period information update from user?” Siler, Figure 4, ¶0029. Thus, Siler disclosed that the first computer system (in the form of web server 113) received information from the user client 101 after a predetermined time period elapses.

232. Siler disclosed that the information that the user client 101 sends periodically to the web server 113 included the session identifier. Siler, ¶0028. Siler says that the information “preferably includes the user identifier, the session identifier and the stream identifier.” Siler, ¶0028. As explained previously in this declaration, the session identifier is the identifier data sent from the first computer system (in the form of web server 113) to the user client 101.

f. Claim 1.e: “storing data indicative of the received at least portion of the identifier data using the first computer system;”

233. It is my opinion that Siler discloses this feature. Siler discloses that, when the web server 113 receives the periodic tracking information from the user client 101, it passes the information to the session tracking logic 123. Siler, ¶0029. Siler discloses that the session tracking logic 123 performs “updating the session record for the particular user.” Siler, Figure 4, ¶0029. As explained previously in

this declaration, a person of ordinary skill in the field would have understood that this update of the session record would entail storing the received information in the session records database 125. Siler, Figure 1, ¶0026. As such, a person of ordinary skill in the field would have understood that the first computer system (in the form of web server 113) would have stored the tracking information periodically received from the user client 101.

- g. Claim 1.f: “wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from the first computer system directly to the user’s computer independent of the first computer system;”**

234. It is my opinion that Siler discloses this feature. Siler discloses that the streaming server 105 transmits a data stream “through packet network 103 to the client computer.” Siler, ¶0017. A person of ordinary skill in the field would have understood from this description and the illustration in the figures that the streaming server 105 transmitted the streaming content directly to the user client 101, without passing that content through either of the web servers 113 or 117. Siler, Figure 1, ¶0017-0020.

235. A person of ordinary skill in the field as of August 2008 would have understood that the streaming server could be provided as a distinct “computer system” from the computer system embodied by the web servers 113 and 117. This is true because Siler describes the web servers 113 and 117 as performing

distinct functions from streaming server 105. Siler, ¶¶0017-0020, 0023-0030. In essence, the web servers perform the setup and tracking of the streaming content, but leave the actual streaming of the content to the streaming server 105. Siler, ¶¶0024-0030. A person of ordinary skill in the field would have understood, based on these differing functions, that the streaming server could be operated by one entity, while the web servers could be operated by a different entity.

236. For example, Siler discloses that the streaming content could be from a television station. Siler, ¶¶0018. In that case, the television station, or perhaps an owner of the television station, may control the streaming server 105. This would have made sense so that the content owner could control the access to the streaming content. A separate content owner (e.g., some other television station or a radio station as Siler suggests) could operate their own streaming servers 105. The web servers then could provide a webpage interface for choosing amongst these different streaming options provided by different streaming servers 105. Siler, ¶¶0023. So a person of ordinary skill in the field would have been motivated to provide the streaming server 105 and the web servers 113 and 117 as separate systems operated and controlled by separate entities, at least in situation like in this example.

- h. Claim 1.g: “wherein the stored data is indicative of an amount of time the digital media presentation data**

is streamed from the second computer system to the user's computer; and"

237. It is my opinion that Siler discloses this feature. As I explained previously, it is my opinion that "an amount of time" data was "streamed" should be interpreted to mean "an amount of time" the data was "transferred via a technique such that the data can be processed as a substantially steady or continuous sequence." But as also explained previously, I will analyze this feature under the alternative interpretation that "an amount of time" the data was "streamed" means "an amount of time" the data was "presented" on the user's computer.

238. If "streamed" means "transferred via a technique such that the data can be processed as a substantially steady or continuous sequence," then Siler disclosed this feature. Siler disclosed that the tracking information stored by the web server 113 indicates "which users are receiving a particular media stream and how long each of the users receives" it. Siler, Abstract. As such, Siler suggests to a person of ordinary skill in the field that the tracking information can include tracking how long the user client 101 received the streaming content. Siler also discloses that the streaming content is received and decoded at the user client for "nearly simultaneous playback." Siler, ¶0015. Siler also discloses that the streaming content can be provided "in real time for immediate streaming." Siler,

¶0019. At least in these situations, the amount of time that the user client 101 or the user thereof “receives” the streaming content is also indicative of how long that streaming content was transferred by streaming server 105 to the user client 101.

239. If “streamed” means “presented,” then Siler disclosed this feature. Siler discloses that the tracking information indicates “how many people are actually listening to the content, as well as when they listened and how long they listened.” Siler, ¶0030. A person of ordinary skill in the field would have understood that “how long [a user] listened” to the streaming content was an indication of how long the streaming content was presented to the user of the user client 101. A person of ordinary skill would have further been motivated by Davis’s suggestion to track the duration of streaming content presentation in order to store such information. Davis, 16:63-17:10. While Siler refers to how long content was “listened” to, Siler’s approach would be equally applicable to how long a user “watched” video content.

i. Claim 1.h: “wherein each stored data is together indicative of a cumulative time the corresponding web page was displayed by the user's computer.”

240. It is my opinion that the system of Siler, when modified based on the teachings of Siler and Davis as described previously in this declaration, would include this feature. As described previously, a person of ordinary skill in the field would have been motivated to include tracking of the duration that a webpage was

displayed on the user client 101, in particular of the webpage that implemented the player application 122, especially when additional advertising such as a banner advertisement was displayed. Davis, 11:34-13:18. As such, Siler's system, as modified based on the teachings of Siler and Davis, would have included this feature.

241. Further, to the extent that this feature requires that the web server 113 store multiple tracking information entries for a single webpage visit, a person of ordinary skill in the field would have recognized that this would result when Siler's system was modified as explained previously in this declaration. Namely, any time that the user of the user client 101 remained on the webpage for a longer period of time than the prescribed time, Siler, Figure 3, ¶0028, the user client 101 would send at least two tracking reports to the web server 113. Hence, the two or more tracking reports stored by the web server 113 would together indicate how long the webpage was displayed on the user client 101.

3. Claim 2: “The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.”

242. It is my opinion that when the system described in Siler was modified as explained previously in this declaration as taught by Siler and Davis, a person of ordinary skill in the art would have been motivated to include this feature in Siler's system. As explained previously in this declaration, Siler disclosed that the user

client 101 sent tracking reports to the web server 113 periodically, according to a prescribed time. Siler, Figure 3-4, ¶0028-0029.

243. Davis disclosed that one way to store tracking information was to increment a counter each time a tracking report was received by the server. Davis, 3:42-44. Davis disclosed that, in order to track the number of times that some particular content, such as an advertisement, had been displayed, the server could increment a stored counter. Davis, 3:42-53. The same approach could be used to track the number of times an advertisement was clicked on. Davis, 3:42-53. Thus, a person of ordinary skill would have recognized that it would be possible to increment a stored counter in web server 113 of Siler in order to store some types of tracking information.

244. A person of ordinary skill in the field would have recognized that this incrementing approach could be used by the web server 113 for storing various types of tracking information. For example, instead of storing multiple tracking records for a single webpage visit, each indicating the same predetermined time, the web server 113 could simply increment a value for “number of intervals” as well as the length of the predetermined time (the “prescribed” time). For instance, instead of storing 10 tracking records, each indicating “15 seconds,” the web server

113 could store one tracking record, which would indicate “number of intervals” as “10” and “length of interval” as “15 seconds.”

245. A person of ordinary skill would have recognized that the web server 113 may still need to store a second tracking record, for example, when the last tracking report for the webpage visit indicated a length of the display of the webpage that was less than the full prescribed time. But a person of ordinary skill would have recognized that this would allow a reduction of the number of tracking records from 11 (continuing the previous example) down to two.

246. A person of ordinary skill would have been motivated to use an incrementing approach as disclosed in Davis for a few reasons.

247. First, Davis already disclosed that this approach was possible, and a person of ordinary skill would have been likely to choose a familiar approach recommended by Davis.

248. Second, use of a counter is something that any undergraduate student in computer science or a like field would be very familiar with. Thus, a person of ordinary skill, who would have at least that level of training, would be motivated to use known, simple solution to the problem.

249. Third, a person of ordinary of ordinary skill in the field would have recognized that the incrementing approach would allow a significant reduction in

the size of the database stored by the web server 113. In the example given just above, the reduction was on the order of a five-fold decrease in the number of records stored. This reduction in storage space would provide some marginal benefit to being able to store more tracking information in the web server 113 using the same server specifications.

250. Fourth, a person of ordinary skill in the field would have recognized that using the incrementing approach would make it easier and quicker to process many of the tracking reports that the web server 113 received from the user client 113. In particular, when the web server 113 received a tracking report from the user client 101, the web server 113 could execute a simple update command on the database to increase the “number of intervals” value. If the tracking database used SQL, for example, a single UPDATE statement could be used to store the new tracking information. On the other hand, storing each tracking report as a separate tracking record would require forming and then inserting a new record into the tracking database each time a tracking report was received. A person of ordinary skill would have recognized that this would have required more logic to perform, would introduce more possibilities for error, and would have consumed more resources on the database.

251. Fifth, a person of ordinary skill would have recognized that storing fewer tracking information records for each webpage visit would allow simpler and quicker aggregation of tracking information when the database was later used by its consumers (e.g., website administrators, advertisers). With the incrementing approach, the program that aggregated tracking information for consumption by the website administrators and advertisers would need to process fewer tracking records from the database, which would make the aggregation process simpler and shorter.

4. Claim 3: “The method of claim 2, wherein the received data is indicative of a temporal cycle passing.”

252. It is my opinion that Siler discloses this feature. As explained previously, the web server 113 received data from the user client 101 each time the “prescribed time” elapsed. Siler, Figures 3-4, ¶0028-0029. As such, when the web server 113 received the tracking information from the user client 101, a person of ordinary skill in the art would have understood that it indicated the “prescribed time” temporal cycle of Figure 3 had passed. Siler, Figures 3-4, ¶0028-0029.

IV. CONCLUSION

253. Based on the foregoing analysis, it is my opinion that claims 1-3 of the '609 Patent would have been obvious to a person of ordinary skill in the field as of August 2008 based on the combination of Davis and Choi.

254. Based on the foregoing analysis, it is my opinion that claims 1-3 of the '609 Patent would have been obvious to a person of ordinary skill in the field as of August 2008 based on the combination of Siler and Davis.

Declaration of Michael Franz
In Support of Petition for *Inter Partes* Review of
U.S. Pat. No. 8,407,609

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001; and further that such willful false statements may jeopardize the validity of the application or any patent issued thereon. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on October 18th, 2019 in Berlin, Germany.



Michael Franz

Appendix B

B-1 (Ex. 1001 in the Petition) U.S. Patent No. 8,407,609 B2 to Turner et al. (“’609 Patent”)

B-2 (Ex. 1003 in the Petition) U.S. Pat. No. 5,796,952 (“Davis”)

B-3 (Ex. 1004 in the Petition) U.S. Pat. Appl. Pub. No. 2003/0236905 A1 (“Choi”)

B-4 (Ex. 1005 in the Petition) U.S. Pat. Appl. Pub. No. 2004/0133467 A1 (“Siler”)

B-5 [MS-WMLOG]: Windows Media Log Data Structure Specification, Microsoft Corporation, June 18, 2008.

B-6 [MS-WMLOG]: Windows Media Log Data Structure Specification, Microsoft Corporation, July 23, 2008.

B-7 U.S. Pat. No. 7,089,259.

B-8 U.S. Pat. Appl. Pub. No. 2007/0192652.

B-9 U.S. Pat. Appl. Pub. No. 2007/0294380.

B-10 U.S. Pat. Appl. Pub. No. 2002/0165849.

B-11 U.S. Pat. 6,108,637.

B-12 U.S. Pat. 7,089,304.

B-13 U.S. Pat. 6,877,007.

B-14 U.S. Pat. 7,310,609.

APPENDIX A

Michael Franz

444 Computer Science Building
University of California, Irvine
Irvine, CA 92697-3435
franz@uci.edu

Major Research Emphases

- Secure and Trustworthy Computing, Critical Cyber-Infrastructure Protection. Mobile code security; secure and efficient mobile program representations; code verification. Language-based security. Information flow; system-level end-to-end security properties. Moving target defenses, automatically generated software diversity, n-variant systems.
- Software Execution Environments. Compilers, virtual machines, and machine code generation and optimization. On-the-fly, feedback-directed and continuous compilation and optimization; binary translation; trace-based compilation. Code generation for embedded systems, heterogeneous architectures, and mobile computing; compiling for low power consumption. Automatic parallelization. Memory management.
- Software Engineering. Software architectures for secure systems; minimizing the trusted code base. Component-oriented programming languages and their implementation. Software reliability and robustness.

Education

Doctor of Technical Sciences, ETH Zürich, Switzerland; February 1994

Dissertation Title: “Code-Generation On-the-Fly: A Key to Portable Software”

Advisor: Niklaus Wirth

Diplomingenieur, ETH Zürich; May 1989

Academic Appointments

- 2016 – present **Chancellor’s Professor**
- 2006 – present *Professor of Computer Science* (with tenure)
- 2001 – 2006 *Associate Professor* (with tenure)
- 1996 – 2001 *Assistant Professor*
Department of Computer Science (since January 2003)
Department of Information and Computer Science (until January 2003)
The Donald Bren School of Information & Computer Sciences
University of California, Irvine
- 2007 – present *Professor of Electrical Engineering & Computer Science* (by courtesy)
Department of Electrical Engineering & Computer Science
The Henry Samueli School of Engineering
University of California, Irvine
- 1994 – 1995 *Senior Research Associate (“Oberassistent”) and Lecturer*
Institut für Computersysteme
ETH Zürich, Switzerland

Visiting Appointments

| | |
|---------------------------------|--|
| since June 2019 | <i>Guest Professor</i> (partially on Sabbatical from UC Irvine) Technical University of Braunschweig, Germany (Host: Prof. Dr. Ina Schäfer) |
| August 2010 – September 2011 | <i>Visiting Professor</i> (on Sabbatical from UC Irvine) ETH Zurich, Switzerland (Host: Prof. Dr. Thomas Gross) |
| January – September 2002 | <i>Visiting Researcher</i> (on Sabbatical from UC Irvine) University of California, Berkeley (Host: Prof. Dr. George Necula) |
| Summer Semester 2000 | <i>Visiting Professor</i> University of Klagenfurt, Austria (Host: Prof. Dr. Laszlo Böszörményi) |
| Summer Semester 1998 | <i>Visiting Professor</i> University of Ulm, Germany (Host: Prof. Dr. Peter Schulthess) |

Major Professional Honors

- *Humboldt Research Award*, Alexander von Humboldt Foundation. This award is granted in recognition of a researcher's entire achievements to date to academics whose fundamental discoveries, new theories, or insights have had a significant impact on their own discipline and who are expected to continue producing cutting-edge achievements in the future; 2018.
- *Fellow, Association for Computing Machinery (ACM)*, "For contributions to just-in-time compilation and optimization and to compiler techniques for computer security;" 2015.
- *Fellow, The Institute of Electrical and Electronics Engineers (IEEE)*, "For contributions to just-in-time compilation and to computer security through compiler-generated software diversity;" 2015.
- *IEEE Computer Society Technical Achievement Award*, 2012, "for pioneering contributions to just-in-time compilation and optimization and significantly advancing Web application technology."
- *University of California, Irvine, Distinguished Mid-Career Faculty Award for Research*, 2010. This is the Academic Senate's highest honor for research. One such award at most is given yearly to an Assistant Professor, one to an Associate or Full Professor Step I-IV (the "Mid-Career Award"), and one to a Professor Step V or higher.
- *National Science Foundation CAREER Award*, 1997.
- *Fulbright Scholarship*, 1989.

Teaching Honors

- *Dean's Award for Graduate Student Mentoring*, Donald Bren School of Information and Computer Sciences, UC Irvine, 2007 and 2016.
- *Outstanding Professor of the Year Award*, Graduating Class of 2007, UC Irvine.

Institutional Affiliations

- Director, *Secure Systems and Software Laboratory*, Donald Bren School of Information and Computer Sciences, UC Irvine; since September 2007.
- Charter Faculty Member, *The California Institute for Telecommunications and Information Technology (Cal-(IT)²*, one of four California Institutes for Science and Technology.
- Charter Faculty Member, *Security Computing and Networking Center (SCoNCe)* (previously named *Center for Cyber-Security and Privacy*), Donald Bren School of Information and Computer Sciences, UC Irvine.

Noteworthy Contributions With Wide Impact

I am the co-inventor (with my former Ph.D. student Andreas Gal) of the “Trace Tree” compilation technique, which has been transitioned successfully from academic research into one of the most widely distributed open-source projects. From version 3.5 (June 2009) onwards, the JavaScript engine in *Mozilla’s Firefox* browser has been based directly on my academic research (see publication C.58).

Furthermore, since version 4.0 (March 2011), the *Firefox* browser additionally contains the “Compartmental Memory Manager” developed in collaboration between my lab and Mozilla (see publication C.69). No fewer than four of my former students with completed Ph.D.s are now employed full-time at Mozilla.

Funding

Current Grants and Awards

- *Office of Naval Research*, N00014-17-1-2782, “Attack Surface Reduction for Binary Programs;” 30th September 2017 – 30th September 2020, \$3,157,799 (**lead PI**. This is a collaborative award with Herbert Bos of Vrije Universiteit Amsterdam, Netherlands. My share of the award is \$2,337,935). PM Dr. Sukarno Mertoguno.
- *United States Air Force & Air Force Research Laboratory*, FA8750-16-C-0260, “Thunderlane Phase II;” 1st September 2018 – 12th December 2019, \$457,672 (**sole PI** on sub-award for \$457,672 from prime contractor Assured Information Security, Inc. / Adam Hovak).
- *DARPA, Cyber Fault-tolerant Attack Recovery (CFAR) Program*, FA8750-15-C-0124, “Robust, Assured Diversity for Software Security (RADSS).” In August of 2017, **award was increased by \$217,597** and the duration extended to the end of March 2019. The modified award now runs 13th May 2015 – 31st March 2019, \$2,199,227 (**sole PI** on sub-award for \$2,199,227 from prime contractor Galois, Inc. / Stephen Magill). PM Dr. John Everett and Dr. Jacob Torrey.
- *National Science Foundation, Secure & Trustworthy Cyberspace (SaTC) Program*, CNS-1619211, “TWC:Small: Hydra—Hybrid Defenses for Resilient Applications: Practical Approaches Towards Defense In Depth;” 1st July 2016 – 30th June 2020, \$499,981 (**sole PI**). PM Dr. Sol J. Greenspan.
- *DARPA, Cyber Fault-tolerant Attack Recovery (CFAR) Program*, FA8750-15-C-0124, “Robust, Assured Diversity for Software Security (RADSS);” 13th May 2015 – 1st November 2018, \$1,975,630 (**sole PI** on sub-award for \$1,975,630 from prime contractor Galois, Inc. / Stephen Magill). PM Dr. John Everett.
- *DARPA, Cyber Fault-tolerant Attack Recovery (CFAR) Program*, FA8750-15-C-0085, “RAVEN;” 5th May 2015 – 31st March 2019, \$702,271 (**sole PI** on sub-award for \$702,271 from prime contractor Apogee Research, LLC / Tiffany Frazier). PM Dr. John Everett.

Past Grants and Awards

- *National Science Foundation, Secure & Trustworthy Cyberspace (SaTC) Program*, CNS-1513837, “ENCORE—ENhanced program protection through COMpiler-REwriter cooperation;” 1st July 2015 – 30th June 2018, \$1,199,953 (**lead PI**. This is a collaborative award with Matthias Payer of Purdue University and Kevin Hamlen of The University of Texas at Dallas. My share of the award is \$619,267.) PM Dr. Sol J. Greenspan.
- *United States Air Force & Air Force Research Laboratory*, FA8750-16-C-0260, “Thunderlane;” 19th September 2016 – 24th May 2017, \$45,000 (**sole PI** on sub-award for \$45,000 from prime contractor Assured Information Security, Inc. / Philip White).
- *National Science Foundation, Computing and Communications Foundations Program*, IIP-1439439, “I-Corps: Hardening Programs Against Cyber Attacks;” 1st June 2014 – 30th November 2015, \$50,000 (**sole PI**). PM Dr. Rathinda Dasgupta.
- *DARPA, I2O Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program & Transformative Apps Program*, D11PC20024, “Defending Mobile Apps Through Automated Software Diversity.” In May of 2014, award was increased by \$247,830 and duration extended to 30th September 2015. The modified award now runs 4th February 2011 – 30th September 2015, \$2,095,432 (**sole PI**). PMs Dr. Howard Shrobe and Dr. Robert Laddaga.
- *DARPA, I2O Mission-Oriented Resilient Clouds (MRC) Program*, N66001124014, “Meta-Circular Software Diversity for Intrusion Tolerant Clouds;” 1st July 2012 – 31st October 2015, \$456,809 (**sole PI** on this sub-award for \$456,809, which is part of a larger project led by Yair Amir awarded to Johns Hopkins University). PMs Dr. Howard Shrobe and Dr. Robert Laddaga.

- *DARPA, I2O Vetting Commodity IT Software and Firmware (VET) Program*, N66001-13-C-4057, “Heterogeneous Compilations for Detection of Malice in Embedded Systems,” 1st February 2015 – 30th June 2015, \$64,999 (**sole PI** on sub-award for \$64,999 from prime contractor Apogee Research, LLC / Tiffany Frazier). PM Dr. Timothy Fraser.
- *DARPA, I2O Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program & Transformative Apps Program*, D11PC20024, “Defending Mobile Apps Through Automated Software Diversity.” In June of 2012, award was increased by \$467,442 and duration extended by an additional year. Modified award now runs 4th February 2011 – 3rd February 2015, \$1,847,602 (**sole PI**). PMs Dr. Howard Shrobe, Dr. Robert Laddaga, and Dr. Mari Maeda.
- *National Science Foundation, Computing and Communications Foundations Program*, CCF-1117162, “SHF: CSR: Small: Fine-Grained Modularity and Reuse of VM Components,” 1st August 2011 – 31st July 2014, \$499,867 (**sole PI**). PM Dr. Bill Pugh.
- *DARPA, Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program & Transformative Apps Program*, D11PC20024, “Defending Mobile Apps Through Automated Software Diversity,” 4th February 2011 – 3rd February 2014, \$1,380,162 (**sole PI**). PMs Dr. Howard Shrobe and Dr. Mari Maeda.
- *National Science Foundation, Trusted Computing Program*, CNS-0905684, “Next-Generation Infrastructure for Trustworthy Web Applications,” 1st September 2009 – 31st August 2012, \$600,000 (**lead PI**, award is split evenly with co-PI C. Flanagan of UC Santa Cruz). PM Dr. Karl Levitt.
- *Samsung Telecommunications America*, Richardson, Texas, Agreement No. 51070, “Fine-Grained Modularity and Reuse of Virtual-Machine Components,” 1st January 2011 – 31st December 2011, \$349,965 (**sole PI**). PM Venky Raju.
- *California MICRO Program* and industrial sponsor *Sun Microsystems, Inc.*, Project No. 07-127, “Trace Compilation for a Server Java Virtual Machine,” 24th August 2007 – 30th June 2009, \$81,500 (\$50,000 gift from sponsor, \$31,500 matching cash contribution from MICRO, waiver of overhead charges applies to the total grant amount; **sole PI**).
- *National Intelligence Community, Enterprise Cyber Assurance Program (NICECAP)*, FA8750-07-2-0085, “Leveraging Parallel Hardware to Detect, Quarantine, and Repair Malicious Code Injection,” 17th May 2007 – 17th August 2009, \$1,020,375 (**sole PI**). PM Dr. Carl Landwehr. (This solicitation drew 265 responses, of which 11, including this one, were funded. Among the 11 funded projects, 4 were from M.I.T. and one each from Carnegie-Mellon, Columbia, Cornell, Stanford, and UT Austin. UC Irvine was the only university in the competition to receive a grant awarded to a sole Principal Investigator.)
- *National Science Foundation, Trusted Computing Program*, CNS-0627747, “MLS-VM: Design and Implementation of a Next-Generation Information-Centric Target Platform for Trusted Internet Computing,” 1st September 2006 – 31st August 2010, \$400,000 (**sole PI**). PM Dr. Helen Gill.
- *National Science Foundation, Embedded and Hybrid Systems Program*, CNS-0615443, “Virtual-Machine Techniques for Resource-Constrained Devices: Reconciling Reliability With Reusability and Low Development Costs in the Embedded Systems Space,” 1st July 2006 – 30th June 2010, \$300,000 (**sole PI**). PM Dr. Helen Gill.
- *United States Homeland Security Advanced Research Projects Agency (HSARPA)*, FA8750-05-2-0216, “Adding Mandatory Access Control to Virtual Machines”, 2nd May 2005 – 1st November 2007, \$312,483 (**sole PI**). PM Dr. Douglas Maughan. (My proposal was the only one of 80 submissions in the category “Vulnerability Prevention” that got funded by DHS. Overall, the Homeland Security solicitation drew 583 responses, of which 17, including this one, were funded.)
- *National Science Foundation, Information Technology Research (ITR)*, CCR-0205712, “Virtual Power for a Wireless Campus: Orchestrated Modeling, Analysis, Composition and Compilation Strategies for Distributed Embedded Systems,” 1st September 2002 – 31st August 2005, \$2,000,796 (**lead PI** with C. Krintz and R. Wolski of UC Santa Barbara). PM Dr. Helen Gill. (Award is split \$500,000 to Franz, Krintz and Wolski each, with a further \$500,000 going to an internal sub-contract at UC Irvine with Senior Personnel P. Chou, N. Dutt, and T. Givargis.)

- *California MICRO Program* and industrial sponsor *Microsoft Research*, Project No. 04-032, “Executing Legacy Machine Code on a Safe Virtual Machine,” 11th August 2004 – 30th June 2005, \$46,881 (waiver of overhead charges applies to the total grant amount; **sole PI**).
- *Deutsche Forschungsgemeinschaft (DFG)* [German National Science Foundation], AM-150/1-3, “SafeTSA: Entwicklung syntaxorientierter Verfahren zur sicheren und effizienten Ausführung von mobilem Code,” 1st March 2004 – 28th February 2006, Euro 140,000 (equal co-PI with W. Amme and W. Rossak of the University of Jena, Germany). (This is a new grant that provides continuing support for an earlier DFG-funded research project listed below.)
- *National Science Foundation, Trusted Computing Program*, CCR-TC-0209163, “Practical Language-Based Security, From the Ground Up,” 1st August 2002 – 31st July 2005, \$300,000 (**sole PI**). PM Dr. Carl Landwehr.
- *DARPA Information Systems Office*, F30602-99-1-0536, “New Approaches to Mobile Code: Reconciling Execution Efficiency With Provable Security,” follow-on effort, 22nd June 2002 – 30th September 2003, additional \$207,632 (**sole PI**). PM Dr. Jaynarayan H. Lala.
- *National Science Foundation, Operating Systems and Compilers Program*, CCR-0105710, “Design and Implementation of Component-Oriented Programming Languages,” 1st July 2001 – 30th June 2004, \$240,000 (**sole PI**). PM Dr. Xiaodong Zhang.
- *Department of Defense, Critical Infrastructure Protection and High Confidence, Adaptable Software (CIP/SW) Research Program of the University Research Initiative*, N00014-01-1-0854, “A Comprehensive Context for Mobile-Code Deployment,” 1st May 2001 – 30th September 2004, \$981,121, (**lead PI** with B. Fleisch of UC Riverside). PMs Frank Deckelman and Dr. Ralph Wachter. (Award is split \$793,201 to Franz and \$187,920 to Fleisch. According to the ONR website, “the competition drew 115 white papers, from which 74 proposals were received. After a thorough evaluation by technical expert teams, 20 of these proposals were selected for funding.”)
- *Deutsche Forschungsgemeinschaft (DFG)* [German National Science Foundation], AM-150/1-1, “SafeTSA: Entwicklung syntaxorientierter Verfahren zur sicheren und effizienten Ausführung von mobilem Code,” 23rd August 2001 – 1st February 2004, Euro 135,000 [corresponding to 270,000 Deutsche Marks] (equal Co-PI with W. Amme and W. Rossak of the University of Jena, Germany).
- *National Science Foundation, Next Generation Software Program*, EIA-9975053, “TMO Based Modeling and Design of Reliable Next-Generation Complex Software,” 15th August 1999 – 14th August 2002, \$550,000 (with K. Kim, Principal Investigator, and P. C.-Y. Sheu, Department of Electrical and Computer Engineering, UC Irvine). PM Dr. Frederica Darema. (\$117,000 of the total allocated to co-PI Franz.)
- *California MICRO Program* and industrial sponsor *Microsoft Research*, Project No. 99-039, “An Infrastructure for Dynamic Optimization at Run-Time,” 2nd August 1999 – 30th June 2000, \$38,000 (waiver of overhead charges applies to the total grant amount; **sole PI**).
- *National Science Foundation, Operating Systems and Compilers Program*, CCR-9901689, “Graph-Based Mobile-Code Representations for High-Performance Portable Software,” 1st September 1999 – 31st August 2002, \$180,000 (**sole PI**). PM Dr. Mukesh Singhal.
- *DARPA Information Systems Office*, F30602-99-1-0536, “New Approaches to Mobile Code: Reconciling Execution Efficiency With Provable Security,” 22nd June 1999 – 21st June 2002, \$720,741 (**sole PI**). PM Dr. Jaynarayan H. Lala.
- *National Science Foundation CAREER Award*, CCR-9701400, “Dynamic Optimization of Software Component Systems,” 1st March 1997 – 28th February 2001, \$205,000 (**sole PI**).

Supplementary Awards

- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CNS-0905684, Summer 2011, \$16,000.

- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CNS-0905684, Summer 2010, \$8,000.
- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CNS-0627747, Summer 2007, \$6,000.
- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CNS-0615443, Summer 2007, \$6,000.
- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CCR-0205712, Summer 2004, \$6,000.
- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CCR-0205712, Summer 2003, \$10,000.
- National Science Foundation, Research Experiences for Undergraduates (REU) Award Supplement for Grant CCR-9701400, Summer 1998, \$5,000.

Unrestricted Gifts

- Oracle Corporation, \$100,000; May 2016.
- Qualcomm Corporation, \$40,000; May 2015.
- Oracle Corporation, \$140,000; August 2014.
- Mozilla Corporation, \$83,000; August 2014.
- Oracle Corporation, \$33,000; September 2013.
- Adobe Corporation, \$25,000; August 2011.
- Google Corporation, \$61,000; June 2011.
- Adobe Corporation, \$35,000; August 2010.
- Adobe Corporation, \$40,000; March 2010.
- Mozilla Corporation, \$85,000; December 2009.
- Sun Microsystems, \$80,000; May 2009.
- Google Corporation, \$50,000; January 2008.
- Mozilla Corporation, \$85,000; May 2007.
- Intel Corporation, \$30,000; April 2006.
- Intel Corporation, \$30,000; June 2005.
- Sun Microsystems Laboratories, \$56,031; September 2004
- Intel Corporation, \$30,000; July 2004.
- Microsoft Research, \$33,183, April 2004.

Other Gifts

- Amazon Corporation., \$18,000 in Amazon Web Services credit; September 2012.

Publications

Awarded Patents

- P.1 M. Franz (lead), W. Amme, and J. von Ronne; *Safe Computer Code Formats And Methods For Generating Safe Computer Code*; United States Patent No. 7,117,488; filed October 2001, issued October 2006.
- P.2 M. Franz (lead), A. Gal, and B. Salamat; *Multi-Variant Parallel Program Execution to Detect Malicious Code Injection*; United States Patent No. 8,239,8367 B1; filed March 2008, issued August 2012.
- P.3 M. Franz (lead), W. Amme, and J. von Ronne; *Safe Computer Code Formats And Methods For Generating Safe Computer Code*; United States Patent No. 8,392,897; filed August 2006, issued March 2013.
- P.4 A. Gal (lead) and M. Franz; *Dynamic Incremental Compiler and Method*; United States Patent No. 8,769,511; filed February 2007, issued July 2014.
- P.5 M. Franz (lead), A. Homescu, S. Brunthaler, and P. Larsen; *Code Randomization for Just-In-Time Compilers*; United States Patent No. 9,250,937; filed November 2014, issued February 2016.

Pending and Provisional Patent Applications

- PA.1 M. Franz (lead) and S. Brunthaler; *Using Code Replicas and Randomized Control Flow to Enhance Software Security*; United States Patent Application No. 14/535,313; filed November 2014 (Provisional Patent Application No. 61/900,842, filed November 2013).
- PA.2 P. Larsen (lead), S. Brunthaler, and M. Franz; *Error Report Normalization*; United States Patent Application No. 15/514,811; filed August 2017 (Provisional Patent Application No. 62/058,485, filed October 2014).

Books

- B.2 P. Larsen, S. Brunthaler, L. Davi, A.-R. Sadeghi, and M. Franz; *Automated Software Diversity*; Morgan & Claypool, San Rafael, California, ISBN 978-1-6270-5734-9 (paperback), ISBN 978-1-6270-5755-4 (ebook); December 2015. doi:10.2200/S00686ED1V01Y201512SPT014
- B.1 M. Franz; *Code-Generation On-the-Fly: A Key to Portable Software*, Doctoral Dissertation No. 10497, ETH Zürich; published in book form by Verlag der Fachvereine, Zürich, ISBN 3-7281-2115-0; March 1994.

Edited Volumes

- E.1 M. Franz and P. Papadimitratos (Eds.); *Trust and Trustworthy Computing (Proceedings of the 9th International Conference, TRUST 2016 Vienna, Austria, August 29–30, 2016)*; Springer, Heidelberg, ISBN 978-3-319-45571-6 (paperback), ISBN 978-3-319-45572-3 (ebook); August 2016. doi:10.1007/978-3-319-45572-3

Peer-Reviewed Book Chapters

- BC.10 S. Crane, A. Homescu, P. Larsen, H. Okhravi, and M. Franz; “Diversity and Information Leaks;” in P. Larsen and A.-R. Sadeghi (Eds.), *The Continuing Arms Race: Code-Reuse Attacks and Defenses*, ACM Books, Vol. 18, Morgan & Claypool Publishers, ISBN 978-1-97000-183-9, pp. 61–81; 2018. doi:10.1145/3129743.3129747
- BC.9 T. Jackson, A. Homescu, S. Crane, P. Larsen, S. Brunthaler, and M. Franz; “Diversifying the Software Stack Using Randomized NOP Insertion;” in S. Jajodia, A. K. Ghosh, V. S. Subrahmanian, V. Swarup, C. Wang, X. S. Wang (Eds.), *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*, Springer Advances in Information Security, Vol. 100, ISBN 978-1-4614-5415-1, pp. 151–174; 2013. doi:DOI 10.1007/978-1-4614-5416-8_8

- BC.8 T. Jackson, B. Salamat, A. Homescu, K. Manivannan, G. Wagner, A. Gal, S. Brunthaler, Ch. Wimmer, and M. Franz; “Compiler-Generated Software Diversity;” in S. Jajodia, A.K. Ghosh, V. Swarup, C. Wang, and X.S. Wang (Eds.), *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*; Springer, ISBN 978-1-4614-0976-2, pp. 77–98; September 2011. doi:10.1007/978-1-4614-0977-9_4
- BC.7 M. Franz, W. Amme, M. Beers, N. Dalton, P.H. Fröhlich, V. Haldar, A. Hartmann, P. S. Housel, F. Reig, J. von Ronne, Ch.H. Stork, and S. Zhenochin; “Making Mobile Code Both Safe And Efficient;” in J. Lala (Ed.), *Foundations of Intrusion Tolerant Systems*; IEEE Computer Society Press, ISBN 0-7695-2057-X, pp. 337–356; December 2003. doi:10.1109/FITS.2003.1264941 (Expanded version of conference paper C.15)
- BC.6 M. Franz; “A Fresh Look At Low-Power Mobile Computing;” in L. Benini, M. Kandemir, J. Ramanujam (Eds.), *Compilers and Operating Systems for Low Power*; Kluwer Academic Publishers, Boston, ISBN 1-4020-7573-1, pp. 209–220; September 2003. doi:10.1007/978-1-4419-9292-5_12 (Expanded version of conference paper C.18)
- BC.5 M. Franz; “Safe Code: It’s Not Just For Applets Anymore;” in L. Böszörményi and Peter Schojer (Eds.), *Modular Programming Languages: Proceedings of the Sixth Joint Modular Languages Conference (JMLC 2003)*, Klagenfurt, Austria; Springer Lecture Notes in Computer Science, No. 2789, ISBN 3-540-40796-0; pp. 12–22; August 2003. (Full Text of Invited Keynote Address)
- BC.4 J. von Ronne, A. Hartmann, W. Amme, and M. Franz; “Efficient Online Optimization by Utilizing Offline Analysis and the SafeTSA Representation;” in J. Powers and J. T. Waldron (Eds.), *Recent Advances in Java Technology: Theory, Application, Implementation*; Computer Science Press, Trinity College Dublin, Dublin, Ireland, ISBN 0-9544145-0-0, pp. 233–241; November 2002. (Expanded version of conference paper C.22)
- BC.3 M. Franz; “Oberon: The Overlooked Jewel;” in L. Böszörményi, J. Gutknecht, G. Pomberger (Eds.), *The School of Niklaus Wirth: The Art of Simplicity*; Morgan Kaufmann, San Francisco; ISBN 1-55860-723-4, pp. 41–53; September 2000.
- BC.2 J. Gutknecht and M. Franz; “Oberon with Gadgets: A Simple Component Framework;” in M. Fayad, D. Schmidt, R. Johnson (Eds.), *Implementing Application Frameworks: Object-Oriented Frameworks at Work*; Wiley, ISBN 0-4712-5201-8, pp. 323–338; September 1999.
- BC.1 M. Franz; “Adaptive Compression of Syntax Trees and Iterative Dynamic Code Optimization: Two Basic Technologies for Mobile-Object Systems;” in J. Vitek and Ch. Tschudin (Eds.), *Mobile Object Systems: Towards the Programmable Internet*; Springer Lecture Notes in Computer Science, No. 1222, ISBN 3-540-62852-5, pp. 263–276; February 1997. doi:10.1007/3-540-62852-5_19

Strongly Reviewed Journal & Magazine Articles

*Note: Several conference proceedings have appeared as “special issues” of journals. My contributions to such journal special issues that contain regular conference proceedings are **not** included in this section but are listed under “conference papers” below.*

- J.36 B. Belleville, W. Shen, S. Volckaert, A.M. Azab, and M. Franz; “KALD: Detecting Direct Pointer Disclosure Vulnerabilities;” accepted for publication in *IEEE Transactions on Dependable and Secure Computing (TDSC)*; 2019. doi:10.1109/TDSC.2019.2915829
- J.35 M. Franz; “Making Multivariant Programming Practical and Inexpensive;” in *IEEE Security and Privacy*, Vol. 16, No. 3, pp. 90–94; May 2018. doi:10.1109/MSP.2018.2701161
- J.34 N. Burow, S.C. Carr, J. Nash, P. Larsen, M. Franz, S. Brunthaler, and M. Payer; “Control-Flow Integrity P^3 : Protection, Precision, and Performance;” in *ACM Computing Surveys (CSUR)*, Vol. 50, No. 1, Article No. 16; April 2017. doi:10.1145/3054924
- J.33 A. Homescu, T. Jackson, S. Crane, S. Brunthaler, P. Larsen, and M. Franz; “Large-scale Automated Software Diversity—Program Evolution Redux;” in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Vol. 14, No. 2, March/April 2017. doi:10.1109/TDSC.2015.2433252

- J.32 G. Wagner, P. Larsen, S. Brunthaler, and M. Franz; “Thinking Inside the Box: Compartmentalized Garbage Collection;” in *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 38, No. 3, Article No. 9; May 2016. doi:10.1145/2866576
- J.31 P. Larsen, A. Homescu, S. Brunthaler, and M. Franz; “Automatic Software Diversity;” in *IEEE Security and Privacy*, Vol. 13, No. 2, pp. 30-37; March 2015. doi:10.1109/MSP.2015.23
- J.30 G. Savrun-Yeniceri, W. Zhang, H. Zhang, E. Seckler, C. Li, S. Brunthaler, P. Larsen, and M. Franz; “Efficient Hosted Interpreters on the JVM;” in *ACM Transactions on Architecture and Code Optimization (TACO)*, Vol. 11, No. 1, Article No. 9; February 2014. doi:10.1145/2532642
- J.29 P. Larsen, S. Brunthaler, and M. Franz; “Security through Diversity: Are We There Yet?;” in *IEEE Security and Privacy*, Vol. 12, No. 2, pp. 28–35; March 2014. doi:10.1109/MSP.2013.129
- J.28 Ch. Kerschbaumer, E. Hennigan, P. Larsen, S. Brunthaler, and M. Franz; “Information Flow Tracking meets Just-In-Time Compilation;” in *ACM Transactions on Architecture and Code Optimization (TACO)*, Vol. 10, No. 4, Article No. 38; December 2013. doi:10.1145/2541228.2555295
- J.27 G. Wagner, A. Gal, and M. Franz; “Slimming a Java Virtual Machine by way of Cold Code Removal and Optimistic Partial Program Loading;” in *Science of Computer Programming*, Vol. 76, No. 11, pp. 1037–1053; November 2011. doi:10.1016/j.scico.2010.04.008 (Expanded version of conference paper C.53)
- J.26 B. Salamat, T. Jackson, G. Wagner, Ch. Wimmer, and M. Franz; “Run-Time Defense Against Code Injection Attacks Using Replicated Execution;” in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Vol. 8, No. 4; July 2011. doi:10.1109/TDSC.2011.18
- J.25 W. Amme, J. von Ronne, Ph. Adler, and M. Franz; “The Effectiveness of Producer-Side Machine-Independent Optimizations for Mobile Code;” in *Software—Practice and Experience*, Vol. 39, No. 10, pp. 923–946; July 2009. doi:10.1002/spe.v39:10 (Expanded version of conference paper C.40)
- J.24 E. Yardimci and M. Franz; “Mostly Static Program Partitioning of Binary Executables;” in *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 31, No. 5, Article No. 17; June 2009. doi:10.1145/1538917.1538918
- J.23 A. Gal, Ch.W. Probst, and M. Franz; “Java Bytecode Verification via Static Single Assignment Form;” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 30, No. 4, Article No. 21, pp. 1–21; July 2008. doi:10.1145/1377492.1377496
- J.22 E. Yardimci and M. Franz; “Dynamic Parallelization of Binary Executables on Hierarchical Platforms;” *The Journal of Instruction-Level Parallelism*, Vol. 10, Paper 6, ISSN 1942-9525, pp. 1–24; June 2008. <http://www.jilp.org/vol10/v10paper6.pdf> (Expanded version of conference paper C.41)
- J.21 M. Franz; “Containing the Ultimate Trojan Horse;” *IEEE Security and Privacy*, Vol. 5, No. 4, pp. 52–56; July 2007. doi:10.1109/MSP.2007.77
- J.20 W. Amme, J. von Ronne, and M. Franz; “SSA-Based Mobile Code: Implementation and Empirical Evaluation;” *ACM Transactions on Architecture and Code Optimization (TACO)*, Vol. 4, No. 2, Article No. 13; June 2007. doi:10.1145/1250727.1250733
- J.19 V. Venkatachalam, M. Franz, and Ch.W. Probst; “A New Way Of Estimating Compute Boundedness And Its Application To Dynamic Voltage Scaling;” *International Journal of Embedded Systems (IJES)*, Vol. 3, No. 1/2, pp. 17–30; 2007. doi:10.1504/IJES.2007.016030
- J.18 V. Venkatachalam and M. Franz; “Power Reduction Techniques For Microprocessor Systems;” *ACM Computing Surveys (CSUR)*, Vol. 37, No. 3, pp. 195–237; September 2005. doi:10.1145/1108956.1108957
- J.17 M. Franz, D. Chandra, A. Gal, V. Haldar, Ch.W. Probst, F. Reig, and N. Wang; “A Portable Virtual Machine Target For Proof-Carrying Code;” *Science of Computer Programming*, (Special Issue on Interpreters, Virtual Machines, and Emulators), Vol. 57, No. 3, pp. 275–294; September 2005. doi:10.1016/j.scico.2004.09.001 (Expanded version of conference paper C.28)

- J.16 M. Franz, P.H. Fröhlich, and A. Gal; “Supporting Software Composition at the Programming-Language Level;” *Science of Computer Programming*, (Special Issue on New Software Composition Concepts), Vol. 56, Nos. 1–2, pp. 41–57; April 2005. doi:10.1016/j.scico.2004.11.004
- J.15 W. Amme and M. Franz; “Effiziente Codegenerierung für mobilen Code;” *Informatik-Spektrum*, Vol. 26, No. 4, pp. 237–246; August 2003. doi:10.1007/s00287-003-0317-1
- J.14 T. Kistler and M. Franz; “Continuous Program Optimization: A Case Study;” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 25, No. 4, pp. 500–548; July 2003. doi:10.1145/778559.778562
- J.13 T. Kistler and M. Franz; “Continuous Program Optimization: Design and Evaluation;” *IEEE Transactions on Computers*, Vol. 50, No. 6, pp. 549–566; June 2001. doi:10.1109/12.931893
- J.12 T. Kistler and M. Franz; “Automated Data-Member Layout of Heap Objects to Improve Memory-Hierarchy Performance;” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 22, No. 3, pp. 490–505; May 2000. doi:10.1145/353926.353937
- J.11 T. Kistler and M. Franz; “A Tree-Based Alternative to Java Byte-Codes;” *International Journal of Parallel Programming*, Vol. 27, No. 1, pp. 21–34; February 1999. doi:10.1023/A:1018740018601 (Expanded version of conference paper C.05)
- J.10 M. Franz; “The Java Virtual Machine: A Passing Fad?;” *IEEE Software*, Vol. 15, No. 6, pp. 26–29; November 1998. doi:10.1109/52.730834
- J.09 M. Franz; “Open Standards Beyond Java: On the Future of Mobile Code for the Internet;” *Journal of Universal Computer Science (j.uics)*, Vol. 4, No. 5, pp. 521–532; May 1998. doi:10.3217/jucs-004-05-0522 (Expanded version of conference paper C.08)
- J.08 M. Franz; “Java: Anmerkungen eines Wirth-Schülers“ (in German); *Informatik-Spektrum*, Vol. 21, No. 1, pp. 23–26; February 1998. doi:10.1007/s002870050086
- J.07 M. Franz and T. Kistler; “Slim Binaries;” *Communications of the ACM*, Vol. 40, No. 12, pp. 87–94; December 1997. doi:10.1145/265563.265576
- J.06 M. Franz; “The Programming Language Lagoon: A Fresh Look at Object-Oriented;” *Software-Concepts and Tools*, Vol. 18, No. 1, pp. 14–26; March 1997.
- J.05 M. Franz; “Dynamic Linking of Software Components;” *IEEE Computer*, Vol. 30, No. 3, pp. 74–81; March 1997. doi:10.1109/2.573670
- J.04 M. Brandis, R. Crelier, M. Franz, and J. Templ; “The Oberon System Family;” *Software—Practice and Experience*, Vol. 25, No. 12, pp. 1331–1366; December 1995.
- J.03 M. Franz; “Protocol Extension: A Technique for Structuring Large Extensible Software-Systems;” *Software—Concepts and Tools*, Vol. 16, No. 2, pp. 86–94; July 1995.
- J.02 M. Franz; “The Case for Universal Symbol Files;” *Structured Programming*, Vol. 14, No. 3, pp. 136–147; October 1993.
- J.01 M. Franz; “Emulating an Operating System on Top of Another;” *Software—Practice and Experience*, Vol. 23, No. 6, pp. 677–692; June 1993.

Strongly Reviewed Conference and Workshop Papers

Note: Several conference proceedings have appeared as “special issues” of journals. They are included in this section rather than under “journal articles” above, and for faster identification have been marked with an asterisk. Talks given at conferences are annotated in this section and are not listed again under “presentations” below.

- C.104 D.K. Song, J. Lettner, P. Rajasekaran, Y. Na, S. Volckaert, P. Larsen, and M. Franz; “SoK: Sanitizing for Security;” in *40th IEEE Symposium on Security and Privacy*, San Francisco, California, pp. 187–207; May 2019. doi:10.1109/SP.2019.00010 84 papers accepted out of 673 submissions plus 10 revised papers from the previous year = 12.5%
- C.103 D.K. Song, F. Hetzelt, D. Das, Ch. Spensky, Y. Na, S. Volckaert, G. Vigna, Ch. Kruegel, J.-P. Seifert, and M. Franz; “PeriScope: An Effective Probing and Fuzzing Framework for the Hardware-OS Boundary;” in *2019 Network and Distributed Systems Security Symposium (NDSS 2019)*, Internet Society, ISBN 1-891562-55-X, San Diego, California; February 2019. doi:10.14722/ndss.2019.23176 89 papers accepted out of 521 submissions = 17%
- C.102 T. Kroes, A. Altinay, J. Nash, Y. Na, S. Volckaert, H. Bos, M. Franz, and Ch. Giuffrida; “BinRec: Attack Surface Reduction Through Dynamic Binary Recovery;” in *2018 Workshop on Forming an Ecosystem Around Software Transformation (FEAST ’18)*, Toronto, Canada, pp. 8–13; October 2018. doi:10.1145/3273045.3273050
- C.101 B. Belleville, H. Moon, J. Shin, D. Hwang, J.M. Nash, S. Jung, Y. Na, S. Volckaert, P. Larsen, Y. Paek, and M. Franz; “Hardware Assisted Randomization of Data;” in M. Bailey, Th. Holz, M. Stamatogiannakis, and S. Ioannidis (Eds.), *21st International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 2018)*, Heraklion, Crete, Greece, Springer Lecture Notes in Computer Science Vol. 11050, ISBN 978-3-030-00469-9, pp. 337–358; September 2018. doi:10.1007/978-3-030-00470-5_16 33 papers accepted out of 145 submissions = 23%
- C.100 J. Lettner, D.K. Song, T. Park, S. Volckaert, P. Larsen, and M. Franz; “PartiSan: Fast and Flexible Sanitization via Run-time Partitioning;” in M. Bailey, Th. Holz, M. Stamatogiannakis, and S. Ioannidis (Eds.), *21st International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 2018)*, Heraklion, Crete, Greece, Springer Lecture Notes in Computer Science Vol. 11050, ISBN 978-3-030-00469-9, pp. 403–422; September 2018. doi:10.1007/978-3-030-00470-5_19 33 papers accepted out of 145 submissions = 23%
- C.99 M. Qunaibit, S. Brunthaler, Y. Na, S. Volckaert and M. Franz; “Accelerating Dynamically-Typed Languages on Heterogeneous Platforms Using Guards Optimization;” in T. Millstein (Ed.), *2018 European Conference on Object-Oriented Programming (ECOOP 2018)*; Amsterdam, Netherlands, LIPIcs–Leibniz International Proceedings in Informatics, Vol. 109, ISBN 978-3-95977-079-8, pp. 16:1–16:29; July 2018. doi:10.4230/LIPIcs.ECOOP.2018.16 26 papers accepted out of 66 submissions = 39%
- C.98 T. Park, J. Lettner, Y. Na, S. Volckaert and M. Franz; “Bytecode Corruption Attacks Are Real—And How To Defend Against Them;” in C. Giuffrida, S. Bardin, and G. Blanc (Eds.), *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2018)*, Paris, France, ISBN 978-3-319-93410-5, pp. 326–348; June 2018. doi:10.1007/978-3-319-93411-2_15 18 papers accepted out of 59 submissions = 30%
- C.97 P. Biswas, A. Di Federico, S.A. Carr, P. Rajasekaran, S. Volckaert, Y. Na, M. Franz, and M. Payer; “Venerable Variadic Vulnerabilities Vanquished;” in *USENIX Security 2017*, Vancouver, British Columbia, ISBN 978-1-931971-40-9, pp. 186–198; August 2017. 85 papers accepted out of 522 submissions = 16%
- C.96 S. Volckaert, B. Coppens, B. De Sutter, K. De Bosschere, P. Larsen, and M. Franz; “Taming Parallelism in a Multi-Variant Execution Environment;” in *EuroSys 2017*, Belgrade, Serbia, ISBN 978-1-4503-4938-3, pp. 270–285; April 2017. doi:10.1145/3064176.3064178 41 papers accepted out of 182 valid submissions = 22%
- C.95 R. Rudd, R. Skowrya, D. Bigelow, V. Dedhia, Th. Hobson, S. Crane, Ch. Liebchen, P. Larsen, L. Davi, M. Franz, A.-R. Sadeghi, and H. Okhravi; “Address Oblivious Code Reuse: On the Effectiveness of Leakage Resilient Diversity;” in *2017 Network and Distributed System Security Symposium (NDSS 2017)*, Internet Society, ISBN 1-891562-46-0, San Diego, California; February 2017. doi:10.14722/ndss.2017.23477 68 papers accepted out of 423 submissions = 16%
- C.94 S. Volckaert, B. Coppens, A. Voulimeneas, A. Homescu, P. Larsen, B. De Sutter, and M. Franz; “Secure and Efficient Application Monitoring and Replication;” in *2016 USENIX Annual Technical Conference (ATC 2016)*, Denver, Colorado, ISBN 978-1-931971-30-0, pp. 167–179; June 2016. 47 papers accepted out of 266 submissions = 17.6%
- C.93 J. Lettner, B. Kollenda, A. Homescu, P. Larsen, F. Schuster, L. Davi, A.-R. Sadeghi, T. Holz, and M. Franz; “Subversive-C: Abusing and Protecting Dynamic Message Dispatch;” in *2016 USENIX Annual Technical Conference (ATC 2016)*, Denver, Colorado, ISBN 978-1-931971-30-0, pp. 209–221; June 2016. 47 papers accepted out of 266 submissions = 17.6%

- C.92 K. Braden, S. Crane, L. Davi, M. Franz, P. Larsen, Ch. Liebchen, and A.-R. Sadeghi; “Leakage-Resilient Layout Randomization for Mobile Devices;” in *2016 Network and Distributed System Security Symposium (NDSS 2016)*, Internet Society, ISBN 1-891562-41-X, San Diego, California; February 2016. doi:10.14722/ndss.2016.23364
60 papers accepted out of 389 submissions = 15.4%
- C.91 S. Crane, S. Voleckaert, F. Schuster, Ch. Liebchen, P. Larsen, L. Davi, A.-R. Sadeghi, T. Holz, B. De Sutter, and M. Franz; “It’s a TRAP: Table Randomization and Protection against Function Reuse Attacks;” in *22nd ACM Conference on Computer and Communications Security (CCS 2015)*, Denver, Colorado, ACM Press, ISBN 978-1-4503-3832-5, pp. 243–255; October 2015. doi:10.1145/2810103.2813682 128 papers accepted out of 646 submissions = 19.4%
- C.90 M. Conti, S. Crane, L. Davi, M. Franz, P. Larsen, Ch. Liebchen, M. Negro, M. Qunaibit, and A.-R. Sadeghi; “Losing Control: On the Effectiveness of Control-Flow Integrity under Stack Attacks;” in *22nd ACM Conference on Computer and Communications Security (CCS 2015)*, Denver, Colorado, ACM Press, ISBN 978-1-4503-3832-5, pp. 952-963; October 2015. doi:10.1145/2810103.2813671 128 papers accepted out of 646 submissions = 19.4%
- C.89 G. Savrun-Yeniceri, M. L. Van de Vanter, P. Larsen, S. Brunthaler, and M. Franz; “Efficient and Generic Event-based Profiler Framework for Dynamic Languages;” in *2015 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools (PPPJ’15)*, Melbourne, Florida, ACM Press, ISBN 978-1-4503-3712-0, pp. 102–112; September 2015. doi:10.1145/2807426.2807435
- C.88 C. Stancu, Ch. Wimmer, S. Brunthaler, P. Larsen, and M. Franz; “Safe and Efficient Hybrid Memory Management for Java;” in *International Symposium on Memory Management 2015 (ISMM’15)*, Portland, Oregon, ACM Press, ISBN 978-1-4503-3589-8, pp. 81-92; June 2015. doi:10.1145/2754169.2754185
- C.87 S. Crane, Ch. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, and M. Franz; “Readactor: Practical Code Randomization Resilient to Memory Disclosure;” in *36th IEEE Symposium on Security and Privacy*, San Jose, California; May 2015. doi:10.1109/SP.2015.52 55 papers accepted out of 407 submissions = 13.5%
- C.86 S. Crane, A. Homescu, S. Brunthaler, P. Larsen, and M. Franz; “Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity;” in *2015 Network and Distributed System Security Symposium (NDSS 2015)*, San Diego, California; February 2015. doi:10.14722/ndss.2015.23264 51 papers accepted out of 302 submissions = 16.9%
- C.85 V. Mohan, P. Larsen, S. Brunthaler, K. Hamlen, and M. Franz; “Opaque Control-Flow Integrity;” in *2015 Network and Distributed System Security Symposium (NDSS 2015)*, San Diego, California; February 2015. doi:10.14722/ndss.2015.23271 51 papers accepted out of 302 submissions = 16.9%
- C.84 M. Murphy, P. Larsen, S. Brunthaler, and M. Franz; “Software Profiling Options and Their Effects on Security Based Code Diversification;” in *First ACM Workshop on Moving Target Defense (MTD 2014)*, Scottsdale, Arizona, ACM Press, ISBN 978-1-4503-3150-0, pp. 87–96; November 2014. doi:10.1145/2663474.2663485
- C.83 W. Zhang, P. Larsen, S. Brunthaler, and M. Franz; “Accelerating Iterators in Optimizing AST Interpreters;” in *2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications (OOPSLA 2014)*, Portland, Oregon, ACM Press, ISBN 978-1-4503-2585-1, pp. 727–743; October 2014. doi:10.1145/2660193.2660223 52 papers accepted out of 186 submissions = 28%
- C.82 C. Stancu, Ch. Wimmer, S. Brunthaler, P. Larsen, and M. Franz; “Comparing Points-to Static Analysis with Runtime Recorded Profiling Data;” in *2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools (PPPJ 2014)*, Cracow, Poland, ACM Press, ISBN 978-1-4503-2926-2, pp. 157–168; September 2014. doi:10.1145/2647508.2647524
- C.81 P. Larsen, A. Homescu, S. Brunthaler, and M. Franz; “SoK: Automated Software Diversity;” in *35th IEEE Symposium on Security and Privacy*, San Jose, California, IEEE, ISBN 978-1-4799-4686-0, pp. 276-291; May 2014. doi:10.1109/SP.2014.25 44 papers accepted out of 334 submissions = 13%
- C.80 Ch. Kerschbaumer, E. Hennigan, P. Larsen, S. Brunthaler, and M. Franz; “Information Flow Tracking meets Just-In-Time Compilation;” in *High Performance and Embedded Architecture and Compilation Conference (HiPEAC 2014)*, Vienna, Austria; January 2014. doi:10.1145/2541228.2555295

- C.79 Ch. Kerschbaumer, E. Hennigan, P. Larsen, S. Brunthaler, and M. Franz; “CrowdFlow: Efficient Information Flow Security;” *16th Information Security Conference (ISC 2013)*, Dallas, Texas; November 2013. Springer Lecture Notes in Computer Science, Vol. 7807, ISBN 978-3-319-27658-8, pp. 321–340; December 2015. doi:10.1007/978-3-319-27659-5
- C.78 A. Homescu, P. Larsen, S. Brunthaler, and M. Franz; “librando: Transparent Code Randomization for Just-in-Time Compilers;” in *20th ACM Conference on Computer and Communications Security (CCS 2013)*, Berlin, Germany, ACM Press, ISBN 978-1-4503-2477-9, pp. 993–1004; November 2013. doi:10.1145/2508859.2516675
105 papers accepted out of 530 submissions = 19.8%
- C.77 G. Savrun-Yeniceri, W. Zhang, H. Zhang, C. Li, S. Brunthaler, P. Larsen, and M. Franz; “Efficient Interpreter Optimizations for the JVM;” in *2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools (PPPJ’13)*, Stuttgart, Germany, ACM Press, ISBN 978-1-4503-2111-2, pp. 113–123; September 2013. doi:10.1145/2500828.2500839
- C.76 S. Crane, P. Larsen, S. Brunthaler, and M. Franz; “Booby Trapping Software;” in *2013 New Security Paradigms Workshop (NSPW 2013)*, Banff, Canada, ACM Press, ISBN 978-1-4503-2582-0, pp. 95–106; September 2013. doi:10.1145/2535813.2535824
- C.75 E. Hennigan, Ch. Kerschbaumer, P. Larsen, S. Brunthaler, and M. Franz; “First-Class Labels: Using Information Flow to Debug Security Holes;” in M. Huth, N. Asokan, S. Capkun, I. Flechais, and L. Coles-Kemp (Eds.), *Trust and Trustworthy Computing, 6th International Conference (TRUST 2013)*, London, United Kingdom, Springer Lecture Notes in Computer Science, Vol. 7904, ISBN 978-3-642-38907-8, pp. 151–168; June 2013. doi:10.1007/978-3-642-38908-5_12
- C.74 Ch. Kerschbaumer, E. Hennigan, P. Larsen, S. Brunthaler, and M. Franz; “Towards Precise and Efficient Information Flow Control in Web Browsers;” in M. Huth, N. Asokan, S. Capkun, I. Flechais, and L. Coles-Kemp (Eds.), *Trust and Trustworthy Computing, 6th International Conference (TRUST 2013)*, London, United Kingdom, Springer Lecture Notes in Computer Science, Vol. 7904, ISBN 978-3-642-38907-8, pp. 187–195; June 2013. doi:10.1007/978-3-642-38908-5_14
- C.73 A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz; “Profile-guided Automated Software Diversity;” in *2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2013)*, Shenzhen, China, IEEE, ISBN 978-1-4673-5524-7, pp. 204–214; February 2013. doi:10.1109/CGO.2013.6494997
33 papers accepted out of 117 submissions = 28%
- C.72 A. Homescu, M. Stewart, P. Larsen, S. Brunthaler, and M. Franz; “Microgadgets: Size Does Matter In Turing-complete Return-oriented Programming;” in *6th USENIX Workshop on Offensive Technologies (WOOT ’12)*, Bellevue, Washington; August 2012. <https://www.usenix.org/system/files/conference/woot12/woot12-final9.pdf>
- C.71 Ch. Wimmer, S. Brunthaler, P. Larsen, and M. Franz; “Fine-Grained Modularity and Reuse of Virtual Machine Components;” in *11th Annual International Conference on Aspect-Oriented Software Development (AOSD ’12)*, Potsdam, Germany, ACM Press, ISBN 978-1-4503-1092-5, pp. 203–214; March 2012. doi:10.1145/2162049.2162073
- C.70 M. Chang, B. Mathiske, E. Smith, A. Chaudhuri, M. Bebenita, A. Gal, Ch. Wimmer, and M. Franz; “The Impact of Optional Type Information on JIT Compilation Of Dynamically Typed Languages;” in *7th Dynamic Languages Symposium (DLS 2011)*, Portland, Oregon, ACM Press, ISBN 978-1-4503-0939-4, pp. 13–24; October 2011. doi:10.1145/2047849.2047853
- C.69 G. Wagner, A. Gal, Ch. Wimmer, B. Eich, and M. Franz; “Compartmental Memory Management in a Modern Web Browser;” in *International Symposium on Memory Management 2011 (ISMM’11)*, San Jose, California, ACM Press, ISBN 978-1-4503-0263-0; June 2011. doi:10.1145/1993478.1993496
- C.68 T. Jackson, B. Salamat, G. Wagner, Ch. Wimmer, and M. Franz; “On the Effectiveness of Multi-Variant Program Execution for Vulnerability Detection and Prevention;” in *6th International Workshop on Security Measurements and Metrics (MetriSec’10)*, Bolzano-Bozen, Italy, ACM Press, ISBN 978-1-4503-0340-8, Article No. 7; September 2010. doi:10.1145/1853919.1853929

- C.67 M. Franz; “E unibus pluram: Massive-Scale Software Diversity as a Defense Mechanism;” in *2010 Workshop on New Security Paradigms (NSPW’10)*, Concord, Massachusetts, ACM Press, ISBN 978-1-4503-0415-3, pp. 7-16; September 2010. doi:10.1145/1900546.1900550
- C.66 M. Bebenita, M. Chang, K. Manivannan, G. Wagner, M. Cintra, B. Mathiske, A. Gal, Ch. Wimmer, and M. Franz; “Trace-Based Compilation in Execution Environments without Interpreters;” in A. Krall, H. Mössenböck (Eds.), *8th International Conference on the Principles and Practice of Programming in Java 2010 (PPPJ’10)*, Vienna, Austria, ACM Press, ISBN 978-1-4503-0269-2, pp. 59–68; September 2010. doi:10.1145/1852761.1852771
- C.65 K. Manivannan, Ch. Wimmer, and M. Franz; “Decentralized Information Flow Control on a Bare-Metal JVM;” in *Sixth Annual Workshop on Cyber Security and Information Intelligence Research (CSHIRW’10)*, Oak Ridge, Tennessee, ACM Press, ISBN 978-1-4503-0017-9; April 2010. doi:10.1145/1852666.1852738
- C.64 T. Jackson, Ch. Wimmer, and M. Franz; “Multi-Variant Program Execution for Vulnerability Detection and Analysis;” in *Sixth Annual Workshop on Cyber Security and Information Intelligence Research (CSHIRW’10)*, Oak Ridge, Tennessee, ACM Press, ISBN 978-1-4503-0017-9; April 2010. doi:10.1145/1852666.1852708
- C.63 Ch. Wimmer and M. Franz; “Linear Scan Register Allocation on SSA Form;” in *The Eighth International Symposium on Code Generation and Optimization (CGO 2010)*, Toronto, Canada, ACM Press, ISBN 978-1-60558-635-9, pp. 170–179; April 2010. doi:10.1145/1772954.1772979
- C.62 A. Yermolovich, Ch. Wimmer, and M. Franz; “Optimization of Dynamic Languages Using Hierarchical Layering of Virtual Machines;” in *5th Symposium on Dynamic Languages (DLS 2009)*, Orlando, Florida, ACM Press, ISBN 978-1-60558-769-1, pp. 79–88; October 2009. doi:10.1145/1640134.1640147
- C.61 Ch. Wimmer, M. Cintra, M. Bebenita, M. Chang, A. Gal, and M. Franz; “Phase Detection using Trace Compilation;” in *7th International Conference on the Principles and Practice of Programming in Java 2009 (PPPJ 2009)*, Calgary, Alberta, ACM Press, ISBN 978-1-60558-598-7, pp. 172–181; August 2009. doi:10.1145/1596655.1596683
- C.60 Ch. Kerschbaumer, G. Wagner, Ch. Wimmer, A. Gal, Ch. Steger, and M. Franz; “SlimVM: A Small Footprint Java Virtual Machine for Connected Embedded Systems;” in *7th International Conference on the Principles and Practice of Programming in Java 2009 (PPPJ 2009)*, Calgary, Alberta, ACM Press, ISBN 978-1-60558-598-7, pp. 133–142; August 2009. doi:10.1145/1596655.1596678
- C.59 M. Bebenita, M. Chang, A. Gal, and M. Franz; “Stream-Based Dynamic Compilation for Object-Oriented Languages;” in M. Oriol and B. Meyer (Eds.), *Objects, Components, Models and Patterns*, 47th International Conference (TOOLS-EUROPE 2009), Zurich, Switzerland, Springer Lecture Notes in Business Information Processing (LNBIP), Vol. 33, ISBN 978-3-642-02570-9, pp. 77–95; June 2009. doi:10.1007/978-3-642-02571-6.6
- C.58 A. Gal, B. Eich, M. Shaver, D. Anderson, B. Kaplan, G. Hoare, D. Mandelin, B. Zbarsky, J. Orendorff, J. Ruderman, E. Smith, R. Reitmaier, M. R. Haghighat, M. Bebenita, M. Chang, and M. Franz; “Trace-based Just-in-Time Type Specialization for Dynamic Languages;” in *Programming Language Design and Implementation (PLDI 2009)*, Dublin, Ireland, ACM Press, ISBN 978-1-60558-392-1, pp. 465–478; June 2009. doi:10.1145/1542476.1542528
- C.57 B. Salamat, T. Jackson, A. Gal, and M. Franz; “Intrusion Detection Using Parallel Execution and Monitoring of Program Variants in User-Space;” in *EuroSys’09*, Nuremberg, Germany, ACM Press, ISBN 978-1-60558-482-9, pp. 33–46; April 2009. doi:10.1145/1519065.1519071
- C.56 M. Franz; “Information-Flow Aware Virtual Machines: Foundations For Trustworthy Computing;” in *Cyber-security Applications and Technologies Conference for Homeland Security (CATCH 2009)*, Washington, D.C., IEEE Computer Society Publications, ISBN 978-0-7695-3568-5, pp. 91–96; March 2009. doi:10.1109/CATCH.2009.45

- C.55 M. Chang, E. Smith, R. Reitmaier, A. Gal, M. Bebenita, Ch. Wimmer, B. Eich, and M. Franz; “Tracing for Web 3.0: Trace Compilation for the Next Generation Web Applications;” in *2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2009)*, Washington, D.C., ACM Press, ISBN 978-1-60558-375-4, pp. 71–80; March 2009. doi:10.1145/1508293.1508304
- C.54 L. Wang and M. Franz; “Automatic Partitioning of Object-Oriented Programs for Resource-Constrained Mobile Devices with Multiple Distribution Objectives;” in *The 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS’08)*, Melbourne, Victoria, Australia, December 2008. doi:10.1109/ICPADS.2008.84
- C.53 G. Wagner, A. Gal, and M. Franz; “SlimVM: Optimistic Partial Program Loading for Connected Embedded Java Virtual Machines;” in L. Veiga, V. Amaral, N. Horspool, and G. Cabri (Eds.), *Principles and Practice of Programming in Java 2008 (PPPJ 2008)*, Proceedings of the 6th International Conference, Modena, Italy, ACM Press, ISBN 978-1-60558-223-8, pp. 117–126; September 2008. doi:10.1145/1411732.1411749 (Best Paper Award)
- C.52 A. Yermolovich, A. Gal, and M. Franz; “Portable Execution of Legacy Binaries on the Java Virtual Machine;” in L. Veiga, V. Amaral, N. Horspool, and G. Cabri (Eds.), *Principles and Practice of Programming in Java 2008 (PPPJ 2008)*, Proceedings of the 6th International Conference, Modena, Italy, ACM Press, ISBN 978-1-60558-223-8, pp. 63–72; September 2008. doi:10.1145/1411732.1411742
- C.51 A. Noll, A. Gal, and M. Franz; “CellVM: A Homogeneous Virtual Machine Runtime System for a Heterogeneous Single-Chip Multiprocessor;” in *2008 ISCA Workshop on Cell Systems and Applications*, Beijing, China; June 2008.
- C.50 B. Salamat, A. Gal, and M. Franz; “Reverse Stack Execution in a Multi-Variant Execution Environment;” in *2008 DSN Workshop on Compiler and Architectural Techniques for Application Reliability and Security (CATARS’08)*, Anchorage, Alaska; June 2008.
- C.49 B. Salamat, A. Gal, T. Jackson, K. Manivannan, G. Wagner, and M. Franz; “Multi-Variant Program Execution: Using Multi-Core Systems to Defuse Buffer-Overflow Vulnerabilities;” in *2008 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008)*, Barcelona, Spain, IEEE Computer Society Press, ISBN 978-0-7695-3109-0, pp. 843–848; March 2008. doi:10.1109/CISIS.2008.136
- C.48 M. Franz; “Eliminating Trust From Application Programs By Way Of Software Architecture;” in *Software Engineering 2008 (SE 2008)*, Munich, Germany, Lecture Notes in Informatics (LNI) No. 121, GI-Edition, Gesellschaft für Informatik, Bonn, ISBN 978-3-88579-215-4, pp. 112–126; February 2008.
- C.47 M. Franz; “Understanding and Countering Insider Threats In Software Development;” in P. Kropf, M. Beny-oucef, and H. Mili (Eds.), *2008 International Montreal Conference on e-Technologies (MCETECH 2008)*, Montreal, Canada, IEEE Computer Society Publications, ISBN 978-0-7695-3082-6, pp. 81–90; January 2008. doi:10.1109/MCETECH.2008.32
- C.46 D. Chandra and M. Franz; “Fine-Grained Information Flow Analysis and Enforcement in a Java Virtual Machine;” in *23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, Florida, IEEE Computer Society Publications, ISBN 0-7695-3060-5, pp. 463–474; December 2007. doi:10.1109/ACSAC.2007.37
- C.45 M. Bebenita, A. Gal, and M. Franz; “Implementing Fast JVM Interpreters In Java Itself;” in V. Amaral, L. Veiga, L. Marcelino, and H. C. Cunningham (Eds.), *Principles and Practices of Programming in Java, Proceedings of the 5th International Conference (PPPJ 2007)*, Lisbon, Portugal, ACM Press, ISBN 978-1-59593-672-1, pp. 145–154; September 2007. doi:10.1145/1294325.1294345
- C.44 A. Gal, M. Bebenita, and M. Franz; “One Method At A Time Is Quite a Waste of Time;” in *Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS’2007)*, Berlin, Germany, Report No. 2007-5, Technische Universität Berlin, ISSN 1436-9915, pp. 11–16; July 2007.
- C.43 M. Franz, A. Gal, and Ch.W. Probst; “Automatic Generation of Machine Emulators: Efficient Synthesis of Robust Virtual Machines for Legacy Software Migration;” in W.-G. Bleek, J. Raasch, H. Züllighoven (Eds.), *Software Engineering 2007 (SE 2007)*, Hamburg, Germany, Lecture Notes in Informatics (LNI) No. 105, GI-Edition, Gesellschaft für Informatik, Bonn, ISBN 978-3-88579-199-7, pp. 83–94; March 2007.

- C.42 A. Gal, Ch.W. Probst, and M. Franz; “HotpathVM: An Effective JIT Compiler for Resource-Constrained Devices;” in *Second International Conference on Virtual Execution Environments (VEE 2006)*, Ottawa, Canada, ACM Press, ISBN 1-59593-332-6, pp. 144–153; June 2006. doi:10.1145/1134760.1134780
- C.41 E. Yardimci and M. Franz; “Dynamic Parallelization of Binary Executables on Hierarchical Platforms;” in *Computing Frontiers 2006*, Ischia, Italy, ACM Press, ISBN 1-59593-302-6, pp. 127–138; May 2006. doi:10.1145/1128022.1128040
- C.40 Ph. Adler, W. Amme, M. Franz, and J. von Ronne; “Producer-Side Platform-Independent Optimizations and Their Effects on Mobile-Code Performance;” in *The 10th IEEE Annual Workshop on Interaction between Compilers and Computer Architectures (INTERACT-10)*, Austin, Texas; February 2006. 14 submitted, 8 accepted
- C.39 V. Haldar, D. Chandra, and M. Franz; “Dynamic Taint Propagation for Java;” in *Twenty-First Annual Computer Security Applications Conference (ACSAC 2005)*, Tucson, Arizona, IEEE Computer Society Publications, ISBN 0-7695-2461-3, pp. 274–282; December 2005. doi:10.1109/CSAC.2005.21
- C.38 V. Haldar, D. Chandra, and M. Franz; “Practical, Dynamic Information-Flow for Virtual Machines;” in *2nd International Workshop on Programming Language Interference and Dependence (PLID’05)*, London, England; September 2005.
- C.37 A. Gal, Ch.W. Probst, and M. Franz; “Average Case vs. Worst Case Margins of Safety in System Design;” in Ch. F. Hempelmann, V. Raskin (Eds.), *New Security Paradigms Workshop 2005 (NSPW 2005)*, Lake Arrowhead, California, ACM Press, ISBN 1-59593-317-4, pp. 25–32; September 2005. doi:10.1145/1146269.1146279
- C*.36 A. Gal, Ch.W. Probst, and M. Franz; “Structural Encoding of Static Single Assignment Form;” in *4th International Workshop on Compiler Optimization Meets Compiler Verification (COCV’05)*, Edinburgh, Scotland; April 2005. Revised post-conference version published as *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 141, No. 2, pp. 85–102; November 2005. doi:10.1016/j.entcs.2005.02.045
- C*.35 W. Amme, J. von Ronne, and M. Franz; “Quantifying the Benefits of SSA-Based Mobile Code;” in *4th International Workshop on Compiler Optimization Meets Compiler Verification (COCV’05)*, Edinburgh, Scotland; April 2005. Revised post-conference version published as *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 141, No. 2, pp. 103–119; November 2005. doi:10.1016/j.entcs.2005.02.046
- C*.34 A. Gal, Ch.W. Probst, and M. Franz; “Integrated Java Bytecode Verification;” in *First International Workshop on Abstract Interpretation of Object-Oriented Programming Languages (AIOOL’05)*, Paris, France; January 2005. Also published as *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 131, pp. 27–38; May 2005. doi:10.1016/j.entcs.2005.01.020
- C.33 V. Haldar and M. Franz; “Symmetric Behavior-Based Trust: A New Paradigm for Internet Computing;” in Carla Marceau, Simon Foley (Eds.), *New Security Paradigms Workshop 2004 (NSPW 2004)*, White Point, Nova Scotia, ACM Press, ISBN 1-59593-076-0, pp. 79–84; September 2004. doi:10.1145/1065907.1066039 (This paper was one of 4 papers selected for the “Highlights of NSPW 2004” session at *ACSAC 2004*.)
- C.32 J. von Ronne, N. Wang, and M. Franz; “Interpreting Programs in Static Single Assignment Form;” in *ACM SIGPLAN 2004 Workshop on Interpreters, Virtual Machines and Emulators (IVME’04)*, Washington, D.C., pp. 23–30; June 2004. doi:10.1145/1059579.1059585
- C.31 M. Beers, Ch.H. Stork, and M. Franz; “Efficiently Verifiable Escape Analysis;” in M. Odersky (Ed.), *18th European Conference on Object-Oriented Programming (ECOOP 2004)*, Oslo, Norway, Springer Lecture Notes in Computer Science, Vol. 3086, ISBN 3-540-22159-X, pp. 75–95; June 2004. doi:10.1007/b98195
- C.30 V. Haldar, D. Chandra, and M. Franz; “Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing;” in *3rd USENIX Virtual Machine Research & Technology Symposium (VM’04)*, San Jose, California, ISBN 1-931971-20-X, pp. 29–41; May 2004. (Best Paper Award)
- C.29 Ch.W. Probst, A. Gal, and M. Franz; “Code Generating Routers: A Network-Centric Approach to Mobile Code;” in *2003 IEEE 18th Annual Workshop on Computer Communications (CCW’2003)*, Dana Point, California, IEEE Press, ISBN 0-7803-8239-0, pp. 179–186; October 2003.

- C.28 M. Franz, D. Chandra, A. Gal, V. Haldar, F. Reig, and N. Wang; “A Portable Virtual Machine Target For Proof-Carrying Code;” in *ACM SIGPLAN 2003 Workshop on Interpreters, Virtual Machines and Emulators (IVME’03)*, San Diego, California, pp. 24–31; June 2003. doi:10.1145/858570.858573
- C.27 J. von Ronne, A. Hartmann, W. Amme, and M. Franz; “Code Annotation for Safe and Efficient Dynamic Object Resolution;” in *2003 Workshop on Compiler Optimization meets Compiler Verification (COCV 2003)*, Warsaw, Poland, April 2003. doi:10.1016/S1571-0661(05)82597-6
- C.26 A. Gal, M. Franz, and D. Beuche, “Learning from Components: Fitting AOP for System Software;” in *Second AOSD 2003 Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS’2003)*, Boston, Massachusetts; March 2003.
- C.25 V. Haldar, Ch.H. Stork, and M. Franz; “The Source Is The Proof;” in C. Serban, S. Saydjari (Eds.), in *ACM SIGSAC 2002 Workshop on New Security Paradigms (NSPW 2002)*, Virginia Beach, Virginia, ACM Press, ISBN 1-58113-598-X, pp. 69–73; September 2002. doi:10.1145/844102.844114 (This paper was one of 4 papers selected for the “Best of NSPW 2002” session at ACSAC 2002.)
- C.24 V. Haldar and M. Franz; “Towards Trusted Systems, From The Ground Up;” in *Tenth ACM SIGOPS European Workshop: Can We Really Depend On An OS? (EW 2002)*, Saint-Emilion, France, ACM Press, pp. 251–254; September 2002. doi:10.1145/1133373.1133426
- C.23 M. Franz; “Enhancing Class Files: A Migration Path to Better Mobile-Code Representations;” in D. Bakken (Ed.), *DSN Fast Abstracts, International Conference on Dependable Systems and Networks (DSN 2002)*, Washington, D.C., June 2002.
- C.22 J. von Ronne, A. Hartmann, W. Amme, and M. Franz; “Efficient Online Optimization by Utilizing Offline Analysis and the SafeTSA Representation;” in *Proceedings of the 2nd Workshop on Intermediate Representation Engineering for Virtual Machines (IRE 2002)*, Dublin, Ireland, June 2002.
- C.21 D. Chandra, Ch. Fensch, W.-K. Hong, L. Wang, E. Yardimci, and M. Franz; “Code Generation at the Proxy: An Infrastructure-Based Approach to Ubiquitous Mobile Code;” in *Fifth ECOOP Workshop on Object-Oriented and Operating Systems (ECOOP-OOSWS 2002)*, Málaga, Spain, June 2002.
- C.20 A. Gal, P.H. Fröhlich, and M. Franz; “An Efficient Execution Model for Dynamically Reconfigurable Component Software;” in *7th International Workshop on Component-Oriented Programming (WCOP 2002)*, Málaga, Spain, June 2002.
- C.19 P.H. Fröhlich and M. Franz; “On Certain Basic Properties of Component-Oriented Programming Languages;” in *First OOPSLA Workshop on Language Mechanisms for Programming Software Components*, Tampa Bay, Florida; October 2001.
- C.18 M. Franz; “A Fresh Look At Low-Power Mobile Computing;” in *Compilers and Operating Systems for Low Power 2001 (COLP 01)*, Barcelona, Spain, pp. 15.1–15.6; September 2001.
- C*.17 Ch.H. Stork, P. S. Housel, V. Haldar, N. Dalton, and M. Franz; “Towards Language Agnostic Mobile Code;” in N. Benton and A. Kennedy (Eds.), *First Workshop on Multi-Language Infrastructure and Interoperability (BABEL’01)*, Florence, Italy; September 2001. Also published as *Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 59, No. 1, pp. 142–157; November 2001. doi:10.1016/S1571-0661(05)80458-X
- C.16 W. Amme, N. Dalton, J. von Ronne, and M. Franz; “SafeTSA: A Type Safe and Referentially Secure Mobile-Code Representation Based on Static Single Assignment Form;” in *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI 2001)*, Snowbird, Utah, pp. 137–147; June 2001. doi:10.1145/378795.378825
- C.15 W. Amme, N. Dalton, P.H. Fröhlich, V. Haldar, P. S. Housel, J. von Ronne, Ch.H. Stork, S. Zhenochin, and M. Franz; “Project transPROse: Reconciling Mobile-Code Security With Execution Efficiency;” in *The Second DARPA Information Survivability Conference and Exhibition (DISCEX II)*, Anaheim, California; IEEE Computer Society Press, ISBN 0-7695-1212-7, pp. II.196–II.210; June 2001. doi:10.1109/DISCEX.2001.932172

- C.14 W. Amme, N. Dalton, M. Franz, and J. von Ronne; “A Type-Safe Mobile Code Representation Aimed At Supporting Dynamic Optimization At The Target Site;” in *Third ACM Workshop on Feedback-Directed and Dynamic Optimization (FDDO-3)*, Monterey, California, December 2000. (Best Paper Award; additionally and independently, the paper’s presentation was one of three simultaneous winners of the Best Presentation Award.)
- C.13 P.H. Fröhlich and M. Franz; “Stand-Alone Messages: A Step Towards Component-Oriented Programming Languages;” in J. Gutknecht and W. Weck (Eds.), *Modular Programming Languages: Proceedings of the Fifth Joint Modular Languages Conference (JMLC 2000)*, Zurich, Switzerland; Springer Lecture Notes in Computer Science, No. 1891, ISBN 3-540-67958-8, pp. 90–103; September 2000. doi:10.1007/10722581_9
- C.12 M. Franz, P.H. Fröhlich, and T. Kistler; “Towards Language Support for Component-Oriented Real-Time Programming;” in *The Fifth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS’99F)*, Monterey, California, November 1999; IEEE Computer Society Press, ISBN 0-7695-0616-X; April 2000. doi:10.1109/WORDSF.1999.842343
- C.11 T. Kistler and M. Franz; “Computing the Similarity of Profiling Data: Heuristics for Guiding Adaptive Optimizations;” in *Proceedings of the Workshop on Profile and Feedback-Directed Optimization*, Paris, France, October 1998.
- C.10 M. Franz; “On the Architecture of Software Component Systems;” in R.N. Horspool (Ed.), *Systems Implementation 2000, (Proceedings of the IFIP TC2 WG2.4 Working Conference on Systems Implementation 2000: Languages, Methods and Tools, Berlin, Germany)*, Chapman & Hall, ISBN 0-412-83530-4, pp. 207–220; February 1998.
- C.09 M. Franz and T. Kistler; “Does Java Have Alternatives?;” in D.J. Richardson and D. Wile (Eds.), *Proceedings of the Third California Software Symposium (CSS’97)*, Irvine, California, pp. 5–10; November 1997.
- C.08 M. Franz; “Beyond Java: An Infrastructure for High-Performance Mobile Code on the World Wide Web;” in S. Lobodzinski and I. Tomek (Eds.), *Proceedings of WebNet 97*, World Conference of the WWW, Internet, and Intranet, Association for the Advancement of Computing in Education; ISBN 1-880094-27-4, pp. 33–38; October 1997. (Best Paper Award)
- C.07 M. Franz; “Run-Time Code Generation as a Central System Service;” in *The Sixth Workshop on Hot Topics in Operating Systems (HotOS VI)*, IEEE Computer Society Press, ISBN 0-8186-7834-8, pp. 112–117; May 1997. doi:10.1109/HOTOS.1997.595192
- C.06 M. Franz; “Toward an Execution Model for Component Software;” in *Proceedings of the First International Workshop on Component-Oriented Programming (WCOP 1996)*, subsequently published as M. Mühlhäuser (Ed.), *Special Issues in Object-Oriented Programming: Workshop Reader of the 10th European Conference on Object-Oriented Programming (ECOOP’96)*, dpunkt Verlag, Heidelberg, ISBN 3-920993-67-5, pp. 144–149; March 1997.
- C.05 T. Kistler and M. Franz; “A Tree-Based Alternative to Java Byte-Codes;” in *Proceedings of the International Workshop on Security and Efficiency Aspects of Java*, Eilat, Israel; January 1997.
- C.04 M. Franz; “Compiler Optimizations Should Pay for Themselves;” in P. Schulthess (Ed.), *Advances in Modular Languages: Proceedings of the Joint Modular Languages Conference*, Universitätsverlag Ulm, ISBN 3-89559-220-X, pp. 111–121; September 1994.
- C.03 M. Franz; “Technological Steps toward a Software Component Industry;” in J. Gutknecht (Ed.), *Programming Languages and System Architectures: Proceedings of the International Conference*, Zurich, Switzerland, Springer Lecture Notes in Computer Science, No. 782, pp. 259–281; March 1994. doi:10.1007/3-540-57840-4_36
- C.02 M. Franz; “Immediate Object-Level Software Reuse on Different Target Architectures using Fast On-The-Fly Code Generation;” in *Position Paper Collection of the Second International Workshop on Software Reusability*, Lucca, Italy; March 1993.
- C.01 M. Franz and S. Ludwig; “Portability Redefined;” in *Proceedings of the Second International Modula-2 Conference*, Loughborough, England, pp. 216–224; September 1991.

Selected Further Conferences, Workshops, and Other Publications

*Note: Technical reports that have subsequently been published as book chapters, conference papers, or journal articles are **not** listed again here.*

- CPC04 E. Yardimci, N. Dalton, Ch. Fensch, and M. Franz; “Azure: A Virtual Machine for Improving Execution of Sequential Programs on Throughput-Oriented Explicitly-Parallel Processors;” in *Proceedings of the 11th International Workshop on Compilers for Parallel Computers (CPC 2004)*, Seeon, Germany, Shaker Verlag, pp. 61–174; July 2004.
- PLOS04 A. Gal, Ch.W. Probst, and M. Franz; “Executing Legacy Applications on a Java Operating System;” in *Proceedings of the ECOOP Workshop on Programming Languages and Operating Systems 2004 (ECOOP-PLOS 2004)*, Oslo, Norway; June 2004.
- TR.04-09 A. Gal, Ch.W. Probst, and M. Franz; *Complexity-Based Denial of Service Attacks on Mobile-Code Systems*; Technical Report No. 04-09, School of Information and Computer Science, University of California, Irvine; April 2004.
- CPC03 N. Dalton, Ch. Fensch, E. Yardimci, and M. Franz; “A Virtual Machine for Improving Native-Code Execution on Explicitly Parallel Processors;” in *Proceedings of the 10th International Workshop on Compilers for Parallel Computers (CPC 2003)*, Amsterdam, The Netherlands, pp. 261–270; January 2003.
- CPC01 J. von Ronne, M. Franz, N. Dalton, and W. Amme; “Compile Time Elimination of Null- and Bounds-Checks;” in *Ninth International Workshop on Compilers for Parallel Computers (CPC 2001)*, Edinburgh, Scotland, pp. 325–334; June 2001.
- TR.98-34 M. Franz and T. Kistler; *Splitting Data Objects to Increase Cache Latency*; Technical Report No. 98-34, Department of Information and Computer Science, University of California, Irvine; October 1998.
- TR.90-142 M. Franz; *MacOberon Reference Manual*; Technical Report No. 142, Departement Informatik, ETH Zürich; November 1990.
- TR.90-141 M. Franz; *The Implementation of MacOberon*; Technical Report No. 141, Departement Informatik, ETH Zürich; October 1990.

Professional Activities

Major Honors and Awards

- *Humboldt Research Award*, Alexander von Humboldt Foundation. This award is granted in recognition of a researcher's entire achievements to date to academics whose fundamental discoveries, new theories, or insights have had a significant impact on their own discipline and who are expected to continue producing cutting-edge achievements in the future. Award of €60,000; 2018.
- *Innovator of the Year Award*, UCI Applied Innovation & The Beall Family Foundation), Award of \$10,000; 2018.
- *Fellow*, Association for Computing Machinery (ACM), "For contributions to just-in-time compilation and optimization and to compiler techniques for computer security;" 2015.
- *Fellow*, The Institute of Electrical and Electronics Engineers (IEEE), "For contributions to just-in-time compilation and to computer security through compiler-generated software diversity;" 2015.
- *Dean's Award for Research*, Donald Bren School of Information and Computer Sciences, UC Irvine, 2015.
- *IEEE Computer Society Technical Achievement Award*, 2012, "for pioneering contributions to just-in-time compilation and optimization and significantly advancing Web application technology."
- *IEEE Orange County Chapter Outstanding Engineer Award*, 2012.
- *University of California, Irvine, Distinguished Mid-Career Faculty Award for Research*, 2010. This is the Academic Senate's highest honor for research. One such award at most is given yearly to an Assistant Professor, one to an Associate or Full Professor Step I-IV (the "Mid-Career Award"), and one to a Professor Step V or higher.
- *Distinguished Scientist*, Association for Computing Machinery (ACM), "Created early mobile code system. Leads key research group on Virtual Machines and Mobile-Code Security. Co-Founder of the ACM Sigplan VEE Conference;" 2006.
- *Senior Member*, The Institute of Electrical and Electronics Engineers (IEEE), 2006.
- *National Science Foundation CAREER Award*, 1997.
- I was awarded a *Fulbright Scholarship* (for graduate study in the United States) in 1989, but subsequently declined this award in order to join the research group of Prof. Niklaus Wirth at ETH Zürich.

Service to the Professional Community: Ongoing

- Program Committee Member, *2020 IEEE Symposium on Security and Privacy ("Oakland")*, San Francisco, California; May 2020.
- Program Committee Member, *GI SICHERHEIT 2020*), Göttingen, Germany; March 2020.
- Program Committee Member, *6th ACM Workshop on Moving Target Defense (MTD 2019)*, London, England; November 2019.
- Program Committee Member, *3rd International Workshop on Software Protection (SPRO-2018)*, London, England; November 2018.
- Program Committee Member, *26th ACM Conference on Computer and Communications Security (ACM CCS 2019)*, London, England; November 2019.
- Member, *IEEE Computer Society Publication Board Best Paper Award (BPA) Committee for IEEE Transactions on Dependable and Secure Computing (TDSC)*; since 2019.
- Member, *IFIP Working Group 11.10 ("Critical Infrastructure Protection")*, 2018 – present.

- Journal Editorial Board Member, *IEEE Transactions on Dependable and Secure Computing (TDSC)*; since March 2015.
- Journal Editorial Board Member, *Software—Practice and Experience (SPE)*; since July 2010.
- Journal Editorial Board Member, *Software-Intensive Cyber-Physical Systems (SICS)*; since October 2009. Prior to 2017, the journal was published under the name *Computer Science—Research and Development (CSR D)*.
- Emeritus Member, *IFIP Working Group 2.4 (“Software Implementation Technology”)*, since July 2018 (previously, I was a Full Voting Member from 2002 – 2018, and before that, an Observer from 1998 – 2002).
- Charter Faculty Member, *Security Computing and Networking Center (SCoNCe)* (previously named *Center for Cyber-Security and Privacy*), Donald Bren School of Information and Computer Sciences, UC Irvine, May 2005 – present.
- Charter Member, *The California Institute for Telecommunications and Information Technology (Cal-(IT)²)*, one of four California Institutes for Science and Technology, December 2000 – present.

Service to the Professional Community: Past

Program Committee Member

- *2019 IEEE Symposium on Security and Privacy (“Oakland”)*, San Francisco, California; May 2019.
- *2018 Dynamic Languages Symposium (DLS18)*, Boston, Massachusetts; November 2018.
- *25th ACM Conference on Computer and Communications Security (ACM CCS 2018)*, Toronto, Ontario, Canada; October 2018.
- *5th ACM Workshop on Moving Target Defense (MTD 2018)*, Toronto, Ontario, Canada; October 2018.
- *2018 Secure Development Conference (SecDev 2018)*, Cambridge, Massachusetts, September/October 2018.
- *19th World Conference on Information Security Applications (WISA 2018)*, Jeju Island, South Korea; August 2018.
- *First Workshop on Software Debloating and Delaying (SALAD ’18)*, Amsterdam, Netherlands; July 2018.
- *38th IEEE International Conference on Distributed Computing Systems (ICDCS 2018)*, Vienna, Austria; June 2018.
- *GI SICHERHEIT 2018*, Constance, Germany; April 2018.
- *2017 ACM/IFIP/USENIX International Middleware Conference (Middleware 2017)*, Las Vegas, Nevada; December 2017.
- *4th ACM Workshop on Moving Target Defense (MTD 2017)*, Dallas, Texas; October 2017.
- *2017 Secure Development Conference (SecDev 2017)*, Cambridge, Massachusetts, September 2017.
- *International Symposium on Engineering Secure Software and Systems (ESSoS’17)*, Bonn, Germany; July 2017.
- *15th International Conference on Applied Cryptography and Network Security (ACNS 2017)*, Kanazawa, Japan; July 2017.
- *37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017)*, Atlanta, Georgia; June 2017.
- *2017 ACM Asia Conference on Computer and Communications Security (ASIACCS 2017)*, Abu Dhabi, UAE; April 2017.
- *8th IEEE International Workshop on Information Forensics and Security (WIFS 2016)*, Abu Dhabi, UAE; December 2016.
- *3rd ACM Workshop on Moving Target Defense (MTD 2016)*, Vienna, Austria; October 2016.
- *2nd International Workshop on Software Protection (SPRO-2016)*, Vienna, Austria; October 2016.

- *23rd ACM Conference on Computer and Communications Security (ACM CCS 2016)*, Vienna, Austria; October 2016.
- *2016 International Conference on Principles and Practices of Programming in Java (PPPJ'2016)*, Lugano, Switzerland; September 2016.
- **Program co-Chair**, *9th International Conference on Trust and Trustworthy Computing (TRUST 2016)*, Vienna, Austria; August 2016.
- *14th International Conference on Applied Cryptography and Network Security (ACNS 2016)*, London, United Kingdom; June 2016.
- *International Symposium on Engineering Secure Software and Systems (ESSoS'16)*, Egham, United Kingdom; March 2016.
- *First IEEE European Symposium on Security and Privacy 2016 (EuroS&P2016)*, Saarbrücken, Germany; March 2016.
- *2nd ACM Workshop on Moving Target Defense (MTD 2015)*, Denver, Colorado; October 2015.
- *22nd ACM Conference on Computer and Communications Security (CCS 2015)*, Denver, Colorado; October 2015.
- *2015 International Conference on Principles and Practices of Programming in Java (PPPJ'2015)*, Melbourne, Florida; September 2015.
- *8th International Conference on Trust and Trustworthy Computing (TRUST 2015)*, Heraklion, Greece; August 2015.
- *1st International Workshop on Software Protection (SPRO-2015)*, Florence, Italy; May 2015.
- *IEEE Workshop on Web 2.0 Security and Privacy 2015 (W2SP'2015)*, San Jose, California; May 2015.
- *International Symposium on Engineering Secure Software and Systems (ESSoS'15)*, Milan, Italy; March 2015.
- *First ACM Workshop on Moving Target Defense (MTD 2014)*, Scottsdale, Arizona; November 2014.
- *2014 New Security Paradigms Workshop (NSPW 2014)*, Victoria, British Columbia, Canada; September 2014.
- *2014 International Conference on Principles and Practices of Programming in Java (PPPJ'2014)*, Krakow, Poland; September 2014.
- *IEEE Workshop on Web 2.0 Security and Privacy 2014 (W2SP'2014)*, San Francisco, California; May 2014.
- *ACM International Conference on Computing Frontiers 2014 (CF 14)*, Cagliari, Italy; May 2014.
- *The Next Generation Malware Attacks and Defense Workshop (NGMAD)*, New Orleans, Louisiana; December 2013.
- *2013 International Conference on Principles and Practices of Programming in Java (PPPJ'2013)*, Stuttgart, Germany; September 2013.
- *2013 New Security Paradigms Workshop (NSPW 2013)*, Banff, Alberta, Canada; September 2013.
- *6th International Conference on Trust and Trustworthy Computing (TRUST 2013)*, London, United Kingdom; June 2013.
- *28th Annual Computer Security Applications Conference (ACSAC 2012)*, Orlando, Florida; December 2012.
- *2012 New Security Paradigms Workshop (NSPW 2012)*, Bertinoro, Italy; September 2012.
- *11th International Conference on Generative Programming and Component Engineering (GPCE 2012)*, Dresden, Germany; September 2012.
- *2012 IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2012)*, Amsterdam, The Netherlands; September 2012.
- *5th International Conference on Trust and Trustworthy Computing (TRUST 2012)*, Vienna, Austria; June 2012.
- *Eighth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2012)*, London, United Kingdom; March 2012.

- *27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, Florida; December 2011.
- **Program co-Chair**, *International Workshop on Programming Language And Systems Technologies for Internet Clients (PLASTIC 2011)*, Portland, Oregon; October 2011.
- *6th Workshop on Programming Languages and Operating Systems (PLOS 2011)*, Cascais, Portugal; October 2011.
- *Third IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT2011)*, Boston, Massachusetts; October 2011.
- *2011 New Security Paradigms Workshop (NSPW 2011)*, Sonoma, California; September 2011.
- *4th International Conference on Trust and Trustworthy Computing (TRUST 2011)*, Pittsburgh, Pennsylvania; June 2011.
- *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI 2011)*, San Diego, California; June 2011.
- *5th International Multidisciplinary Conference on e-Technologies (MCETECH 2011)*, Les Diablerets, Switzerland; January 2011.
- **Program Chair**, *26th Annual Computer Security Applications Conference (ACSAC 2010)*, Austin, Texas; December 2010. 237 submitted papers, 39 accepted.
- *19th ACM/IEEE/IFIP International Conference on Parallel Architectures and Compilation Techniques (PACT 2010)*, Vienna, Austria; September 2010.
- *2010 IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT-10)*, Minneapolis, Minnesota; August 2010.
- *ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems (LCTES 2010)*, Stockholm, Sweden; April 2010.
- *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2010)*, Pittsburgh, Pennsylvania; March 2010.
- *International Symposium on Engineering Secure Software and Systems (ESSoS 2010)*, Pisa, Italy; February 2010.
- **Program co-Chair**, *25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, Hawaii; December 2009.
- *12th Information Security Conference (ISC 2009)*, Pisa, Italy; September 2009.
- *2009 New Security Paradigms Workshop (NSPW 2009)*, Oxford, United Kingdom; September 2009.
- *2009 IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT-09)*, Vancouver, British Columbia, Canada; August 2009.
- *2009 International Conference on Principles and Practices of Programming in Java (PPPJ'2009)*, Calgary, Alberta, Canada; August 2009.
- *47th International Conference on Objects, Models, Components, and Patterns (TOOLS-EUROPE 2009)*, Zurich, Switzerland, June/July 2009.
- *4th Montreal Conference on eTechnologies (MCETECH)*, Ottawa, Canada; May 2009.
- *Compiler Construction 2009 (CC 2009)*, York, United Kingdom; March 2009.
- *2008 IEEE Symposium on Security and Privacy*, Oakland, California; May 2008.
- *2008 Annual IEEE Computer Society/ACM International Symposium on Code Generation and Optimization (CGO 2008)*, Boston, Massachusetts; March 2008.
- *23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, Florida; December 2007.
- *2007 International Conference on Principles and Practices of Programming in Java (PPPJ'2007)*, Monte de Caparica/Lisbon, Portugal; September 2007.

- *New Security Paradigms Workshop (NSPW 2007)*, Washington Valley, New Hampshire; September 2007.
- *Workshop on Linguistic Support for Modern Operating Systems (PLOS 2006)*, October 2006.
- *The Second Workshop on Advances in Trusted Computing (WATC'06 Fall)*, Tokyo, Japan, November-December 2006.
- *Seventh Joint Modular Languages Conference (JMLC 2006)*, Oxford, United Kingdom, September 2006.
- *2006 International Conference on Principles and Practices of Programming in Java (PPPJ'2006)*, Mannheim, Germany, September 2006.
- *New Security Paradigms Workshop (NSPW 2006)*, Dagstuhl, Germany, September 2006.
- *New Security Paradigms Workshop (NSPW 2005)*, Lake Arrowhead, California, September 2005.
- *ECOOP Workshop on Programming Languages and Operating Systems (ECOOP-PLOS 2005)*, June 2005.
- *Third International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2005)*, Edinburgh, Scotland, April 2005.
- *3. Arbeitstagung Programmiersprachen (ATPS 2004) of the German Computer Society (GI)*, Ulm, Germany, September 2004.
- *ECOOP Workshop on Programming Languages and Operating Systems (ECOOP-PLOS 2004)*, Oslo, Norway, June 2004.
- *ACM SIGPLAN 2004 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'04)*, Washington, D.C., June 2004.
- *Second Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2004)*, San Jose, California, March 2004.
- *Third International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2004)*, Barcelona, Spain, March/April 2004.
- *ACM SIGSAC New Security Paradigms Workshop 2003 (NSPW-2003)*, Ascona, Switzerland, September 2003.
- *Sixth Joint Modular Languages Conference (JMLC 2003)*, Klagenfurt, Austria, August 2003.
- *ACM SIGPLAN 2003 Workshop on Interpreters, Virtual Machines and Emulators (IVME'03)*, San Diego, California, June 2003.
- *Second International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2003)*, Warsaw, Poland, April 2003.
- *4th Annual Workshop on Binary Translation (WBT-2002)*, Charlottesville, Virginia, September 2002.
- *Fifth ECOOP Workshop on Object-Oriented and Operating Systems (ECOOP-OOOSWS 2002)*, Málaga, Spain, June 2002.
- *Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002)*, Washington, D.C., April–May 2002.
- *11th International Conference on Compiler Construction (CC'2002)*, Grenoble, France, March 2002.
- *Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, Magdeburg, Germany, May 2001.
- *Fifth Joint Modular Languages Conference (JMLC 2000)*, Zurich, Switzerland, September 2000.
- *Third Workshop on Distributed Communities on the Web (DCW 2000)*, Quebec City, Canada, June 2000.
- *Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000)*, Newport Beach, California, March 2000.
- *European Symposium on Programming (ESOP 2000)*, Berlin, Germany, March/April 2000.
- *Workshop on Binary Translation* (in conjunction with the International Conference on Parallel Architectures and Compilation Techniques, PACT '99), Newport Beach, California, October 1999.

- *ACM Sigplan 1999 Workshop on Compiler Support for System Software (WCSS'99)*, Atlanta, Georgia, May 1999.
- *Fourth California Software Symposium (CSS'98)*, Irvine, California, October 1998.
- *Workshop on Principles of Abstract Machines* (in conjunction with the joint international symposia SAS'98 and PLILP/ALP'98), Pisa, Italy, September 1998.
- *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI'98)*, Montreal, Canada, June 1998.
- *Fourth Joint Modular Languages Conference (JMLC'97)*, Linz, Austria, March 1997.

Session Chair

- *2019 IEEE Symposium on Security and Privacy ("Oakland")*, San Francisco, California; May 2019.
- *25th ACM Conference on Computer and Communications Security (CCS 2018)*, Toronto, Canada; October 2018.
- *38th IEEE International Conference on Distributed Computing Systems (ICDCS 2018)*, Vienna, Austria; June 2018.
- *2018 ACM Asia Conference on Computer and Communications Security (ASIACCS 2018)*, Incheon, South Korea; June 2018.
- *Third IEEE European Symposium on Security and Privacy 2018 (EuroS&P2018)*, London, United Kingdom; April 2018.
- *Usenix Security 2017*, Vancouver, British Columbia; August 2017.
- *23rd ACM Conference on Computer and Communications Security (CCS 2016)*, Vienna, Austria; October 2016.
- *First IEEE European Symposium on Security and Privacy 2016 (EuroS&P2016)*, Saarbrücken, Germany; March 2016.
- *22nd ACM Conference on Computer and Communications Security (CCS 2015)*, Denver, Colorado; October 2015.
- *10th Conference on High Performance and Embedded Architecture and Compilation (HiPEAC 2015)*, Amsterdam, Netherlands; January 2015.
- *6th International Conference on Trust and Trustworthy Computing (TRUST 2013)*, London, United Kingdom; June 2013.
- *22nd International Conference on Compiler Construction (CC 2013)*, Rome, Italy; March 2013.
- *28th Annual Computer Security Applications Conference (ACSAC 2012)*, Orlando, Florida; December 2012.
- *11th International Conference on Generative Programming and Component Engineering (GPCE 2012)*, Dresden, Germany; September 2012.
- *27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, Florida; December 2011.
- *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI 2011)*, San Diego, California; June 2011.
- *26th Annual Computer Security Applications Conference (ACSAC 2010)*, Austin, Texas; December 2010.
- *19th ACM/IEEE/IFIP International Conference on Parallel Architectures and Compilation Techniques (PACT 2010)*, Vienna, Austria; September 2010.
- *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2010)*, Pittsburgh, Pennsylvania; March 2010.
- *23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, Florida; December 2007.
- *Seventh Joint Modular Languages Conference (JMLC 2006)*, Oxford, United Kingdom, September 2006.
- *Invitational Workshop on the Future of Virtual Execution Environments*, Armonk, New York; September 2004
- *New Security Paradigms Workshop (NSPW 2004)*, White Point, Nova Scotia, September 2004.

- *The Fourth IEEE International Conference on Peer-to-Peer Computing (P2P 2004)*, Zurich, Switzerland, August 2004.
- *Southern California Parallel Processing and Computer Architecture Workshop*, Los Angeles, California, May 2004.
- *Sixth Joint Modular Languages Conference (JMLC 2003)*, Klagenfurt, Austria, August 2003.
- *Ninth International Workshop on Compilers for Parallel Computers (CPC 2001)*, Edinburgh, Scotland, June 2001.
- *Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, Magdeburg, Germany, May 2001.
- *Workshop on Binary Translation* (in conjunction with the International Conference on Parallel Architectures and Compilation Techniques, PACT '99), Newport Beach, California, October 1999.
- *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI'98)*, Montreal, Canada, June 1998.
- *Fourth Joint Modular Languages Conference (JMLC'97)*, Linz, Austria, March 1997.

Other Service

- Member, *IEEE Computer Society Fellows Evaluation Committee*; 2019.
- Full Voting Member, *IFIP Working Group 2.4 ("Software Implementation Technology")*, 2002 – 2018 (elevated to Emeritus Member in July 2018).
- Member, *IFIP Working Group 11.3 ("Data and Application Security and Privacy")*, 2008 – 2017.
- Shadow PC Member, *2017 ACM Asia Conference on Computer and Communications Security (ASIACCS 2017)*, Abu Dhabi, UAE; April 2017.
- Member, *IEEE Computer Society Fellows Evaluation Committee*; 2016.
- Nomination Committee Member, MacArthur Fellows Program, *John D. and Catherine T. MacArthur Foundation*; 2016.
- External Review Committee Member, *28th European Conference on Object-Oriented Programming (ECOOP'2014)*, Uppsala, Sweden; July/August 2014.
- External Review Committee Member, *ACM Research Conference on Object-Oriented Programming (OOPSLA 2013)*, Indianapolis, Indiana; October 2013.
- External Review Committee (ERC) Member, *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI 2013)*, Seattle, Washington; June 2013.
- Paper Shepherd, *2011 New Security Paradigms Workshop (NSPW 2011)*, Sonoma, California; September 2011.
- Organization Committee Member and Sponsorship co-chair, *EuroSys 2011*, Salzburg, Austria; March 2011.
- Organizing Committee Member (Student Travel Chair), *Fourteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, Washington, D.C.; March 2009.
- Panels Chair and Conference Committee Member, *24th Annual Computer Security Applications Conference (ACSAC 2008)*, Anaheim, California; December 2008.
- Steering Committee Member, *ACM SIGPLAN/SIGOPS/USENIX International Conference Series on Virtual Execution Environments (VEE)*, 2004 – 2008.
- Local Arrangements Chair, *IFIP WG2.4 Working Meeting*, Arrowhead, California, May 2007.
- Local Arrangements Co-Chair, *New Security Paradigms Workshop (NSPW 2006)*, Dagstuhl, Germany, September 2006.
- **Founding Steering Committee Co-Chair** (with Sam Midkiff of Purdue University), *ACM SIGPLAN/SIGOPS/USENIX International Conference Series on Virtual Execution Environments (VEE)*, September 2004 – June 2005.

- **General Chair**, *ACM SIGPLAN 2004 Workshop on Interpreters, Virtual Machines and Emulators (IVME 2004)*, Washington, D.C., June 2004.
- Observer, *IFIP Working Group 2.4*, February 1998 – November 2002 (elected to full membership on November 14th).
- Tutorials Chair, *ACM Sigplan Conference on Programming Language Design and Implementation (PLDI 2000)*, Vancouver, Canada, June 2000.
- Local Arrangements Co-Chair, *Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000)*, Newport Beach, California, March 2000.
- Executive Committee Member, *The Institute for Software Research at UC Irvine*, July 1999–January 2002.
- Charter Member, *The Institute for Software Research at UC Irvine*, July 1999.
- Session Organizer and Host, *Bay Area Round Table (BART)*, Palo Alto, California, February 1999.
- Executive Committee Member, *Irvine Research Unit in Software (IRUS)* [precursor to The Institute for Software Research], January 1996 – June 1999.
- Swiss Delegate to *IFIP Technical Committee No. 2*, “Software: Theory and Practice,” 1995–1996 term.
- Program Committee Chair, *Oberon Track at the First Joint Annual Conference of the Gesellschaft für Informatik and the Schweizer Informatiker Gesellschaft*, Zürich, September 1995.
- Executive Committee Member, *Special Interest Group on Oberon of the Schweizer Informatiker Gesellschaft*, 1994–1996.
- Organizing Committee Member, *Conference on Programming Languages and System Architectures*, Zürich, March 1994.

Grant Application Review Panel Member

- *National Science Foundation, Program on Software and Trusted Computing (SaTC)*, Arlington, Virginia, January 2016.
- *National Science Foundation, Program on Software and Trusted Computing (SaTC)*, Arlington, Virginia, October 2012.
- *National Science Foundation, Program on Software and Trusted Computing (SaTC)*, Arlington, Virginia, May 2012.
- *National Science Foundation, Program on Computer and Network Systems*, Arlington, Virginia, April 2009.
- *National Science Foundation, Program on Foundations of Computing Processes and Artifacts*, Arlington, Virginia, February 2007.
- *National Science Foundation, CAREER Program in CyberTrust*, Arlington, Virginia, November 2005.
- *National Science Foundation, CAREER Program in Networking and Security*, Arlington, Virginia, November 2003.
- *National Science Foundation, Program in Embedded & Hybrid Systems*, Arlington, Virginia, June 2002.

Invited Keynotes, Presentations and Panels at Conferences

- M. Franz; “Cyber Attacks And Defenses: Trends, Challenges, and Outlook,” *CyberSecurity@KAIST Workshop*, Daejeon, South Korea, June 2018.
- M. Franz; “From Fine Grained Code Diversity to Execute-No-Read: The Cat and Mouse Game Between Attackers and Defenders Continues,” *2nd ACM Workshop on Moving Target Defense (MTD 2015)*, Denver, Colorado; October 2014.

- M. Franz; “Biologically Inspired Software Defenses,” *Fifteenth High Confidence Software and Systems Conference (HCSS 2015)*, Annapolis, Maryland; May 2015.
- M. Franz; “Code Diversity and Biologically Inspired Computer Defenses” (Invited Keynote), *TTI/Vanguard Reprogramming Programming*, Arlington, Virginia; September/October 2014.
- M. Franz; “Software Diversity as a Cyber Defense” (Invited Keynote), *The Next Generation Malware Attacks and Defense Workshop (NGMAD)*, New Orleans, Louisiana; December 2013.
- M. Franz; “Eliminating the Insider Threat in Software Development by Combining Parallelism, Randomization and Checkpointing” (Invited Keynote Address); *Fourth Annual Cyber Security and Information Intelligence Research Workshop (CSIRW’08)*, Oak Ridge National Laboratory, Oak Ridge, Tennessee; May 2008.
- M. Franz; “Security and Privacy in Service Oriented Architectures” (Panelist); *21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC’07)*, Redondo Beach, California; July 2007.
- M. Franz; “Erinnerungen und Ausblicke: Was haben wir gelernt? Und was soll die nächste Generation lernen?” (Invited Panelist); *Tag der Informatik*, ETH Zurich, Switzerland, October 2006.
- M. Franz; “A New Approach to Embedded Java” (Invited Keynote Address); *Mobile Information & Communication Systems, Scientific Conference*, Zurich, Switzerland, October 2006.
- M. Franz; “Pervasive Security” (Panelist); *Software Security Panel, National Science Foundation, Trusted Computing Program, PI Meeting*, Pittsburgh, Pennsylvania; August 2004.
- M. Franz; “Safe Code: It’s Not Just For Applets Anymore” (Invited Keynote Address); *Sixth Joint Modular Languages Conference (JMLC 2003)*, Klagenfurt, Austria, August 2003.
- M. Franz; “Pervasive Security” (Panelist); *Trusted Computing Panel, National Science Foundation, Trusted Computing Program, PI Meeting*, Baltimore, Maryland; August 2003.
- M. Franz; “The Source is The Proof” (Panelist); *NSPW Panel, 18th Annual Computer Security Applications Conference (ACSAC-18)*, Las Vegas, Nevada; December 2002.
- M. Franz; “Extensible Programming: Ein neues Paradigma für die Softwareentwicklung” (Invited Keynote Address, in German); *Moderne Programmierparadigmen*, conference sponsored by Gesellschaft für Informatik, FH Braunschweig-Wolfenbüttel, Germany; October 1994.

Meeting Participation By Invitation († = I gave a presentation, ‡ = my student gave a presentation, * = I presented a poster)

Note: Presentations at conferences with proceedings are documented under “Publications“ above and are not listed again here.

- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Chantilly, Virginia, January 2018.
- † *IFIP WG2.4 Working Meeting*, Essex, Vermont, October 2017.
- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Chantilly, Virginia, May/June 2017.
- † *IFIP WG2.4 Working Meeting*, Dresden, Germany, December 2016.
- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Chantilly, Virginia, November 2016.
- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Arlington, Virginia, April 2016.
- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Arlington, Virginia, January 2016.
- † *M.I.T. Invitational Think-Shop on Multi-Spectrum Metrics for Cyber Defense*, sponsored by the National Science Foundation, Arlington, Virginia; December 2015.
- *DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Arlington, Virginia, November 2015.

- †*DARPA Cyber Fault-tolerant Attack Recovery (CFAR) PI Meeting*, Arlington, Virginia, August 2015.
- †*DARPA Cyber Fault-tolerant Attack Recovery (CFAR) Kick-Off PI Meeting*, Arlington, Virginia, May 2015.
- †*DARPA Joint Clean-Slate Design of Resilient, Secure Hosts (CRASH) & Mission-Oriented Resilient Clouds (MRC) PI Meeting*, Jacksonville, Florida, September 2014.
- †*IFIP WG2.4 Working Meeting*, Asilomar, Pacific Grove, California, February 2014.
- †*DARPA Joint Clean-Slate Design of Resilient, Secure Hosts (CRASH) & Mission-Oriented Resilient Clouds (MRC) PI Meeting*, San Diego, California, January 2014.
- †*M.I.T. Invitational Think-Shop on Multi-Spectrum Metrics for Cyber Defense*, sponsored by the National Science Foundation, Cambridge, Massachusetts; October 2013.
- *Facebook Faculty Summit*, Menlo Park, California, August 2013.
- †*DARPA Joint Clean-Slate Design of Resilient, Secure Hosts (CRASH) & Mission-Oriented Resilient Clouds (MRC) PI Meeting*, Park Ridge, New Jersey, May 2013.
- **National Security Agency, First Annual Science of Security (SoS) Community Meeting*, National Harbor, Maryland, November 2012.
- †*DARPA Joint Clean-Slate Design of Resilient, Secure Hosts (CRASH) & Mission-Oriented Resilient Clouds (MRC) PI Meeting*, San Diego, California, November 2012.
- *DARPA Mission-oriented Resilient Clouds (MRC) Program, PI Meeting*, San Diego, California, October 2012.
- †*DARPA Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program, PI Meeting*, Boston, Massachusetts, May 2012.
- *DARPA Colloquium on Future Directions in Cyber Security*, Arlington, Virginia, November 2011.
- †*DARPA Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program, PI Meeting*, Arlington, Virginia, November 2011.
- †*2nd Army Research Office (ARO) Workshop on Moving Target Defense*, Fairfax, Virginia, October 2011.
- †*DARPA Clean-Slate Design of Resilient, Secure Hosts (CRASH) Program, PI Meeting*, San Jose, California, May 2011.
- *Microsoft Research Faculty Summit*, Redmond, Washington, July 2010.
- †*IFIP WG2.4 Working Meeting*, Berg en Terblijt, Netherlands, January 2010.
- *Networking and Information Technology Research and Development (NITRD) Program, National Cyber Leap Year Summit*, Arlington, Virginia, August 2009.
- †*National Intelligence Community, Enterprise Cyber Assurance Program (NICECAP), PI Meeting*, Washington, D.C., September 2008.
- *Google Faculty Summit*, Mountain View, California, July 2008.
- †*National Intelligence Community, Enterprise Cyber Assurance Program (NICECAP), Reverse Site Visit*, Jessup, Maryland, January 2008.
- †*NCDI Workshop on Game-changing Solutions for Cyber Security* (jointly sponsored by NSF, DHS, IARPA, NSA, ONR, and OSD), College Park, Maryland, November 2007.
- †*National Intelligence Community, Enterprise Cyber Assurance Program (NICECAP), PI Meeting*, Boston, Massachusetts, September 2007.

- *U.S. Department of Energy Workshop on Cyber Security Research Needs for Open Science*, Bethesda, Maryland, July 2007.
- † *IFIP WG2.4 Working Meeting*, Arrowhead, California, May 2007.
- † *National Intelligence Community, Enterprise Cyber Assurance Program (NICECAP), Program Kick-Off Meeting*, Chantilly, Virginia, March 2007.
- † *U.S. Department of Homeland Security, S&T CyberSecurity R&D PI Meeting*, Menlo Park, California, February 2007.
- † *National Science Foundation Safe Computing Workshop*, Albuquerque, New Mexico, November/December 2006.
- † *U.S. Department of Homeland Security, S&T CyberSecurity R&D PI Meeting*, Arlington, Virginia, August 2006.
- † *IFIP WG2.4 Working Meeting*, Glasgow, Scotland, July 2006.
- *The First Workshop on Advances in Trusted Computing*, Tokyo, Japan, March 2006.
- † *U.S. Department of Homeland Security, S&T CyberSecurity R&D PI Meeting*, Menlo Park, California, January 2006.
- *National Science Foundation, Trusted Computing Program, PI Meeting*, Newport Beach, California, September 2005.
- † *U.S. Department of Homeland Security, BAA 04-17, Program Kick-Off Meeting*, Arlington, Virginia, July 2005.
- *Microsoft Academic Days in Silicon Valley*, Mountain View, California, October 2004.
- *Microsoft Research 2004 Faculty Summit*, Redmond, Washington, August 2004.
- † *Southern California Parallel Processing and Computer Architecture Workshop*, Los Angeles, California, May 2004.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, Final Review*, Annapolis, Maryland, May 2004.
- † *IFIP WG2.4 Working Meeting*, Brisbane, Australia, March 2004.
- † *IFIP WG2.4 Working Meeting*, Santa Cruz, California, August 2003.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, PI Meeting*, Ithaca, New York, July 2003.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, Review Meeting*, Arlington, Virginia, June 2003.
- *DARPA Organically Assured and Survivable Information Systems (OASIS) Program, PI Meeting*, Fort Lauderdale, Florida, January 2003.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, PI Meeting*, Irvine, California, January 2003.
- † *IFIP WG2.4 Working Meeting*, Dagstuhl, Germany, November 2002.
- † *DARPA Organically Assured and Survivable Information Systems (OASIS) Program, PI Meeting*, Santa Rosa, California, August 2002.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, PI Meeting*, State College, Pennsylvania, July 2002.
- † *IFIP WG2.4 Working Meeting*, Simon's Town, South Africa, March 2002.
- † *Southern California Parallel Processing and Computer Architecture Workshop*, Irvine, California, February 2002.
- † *ONR Critical Infrastructure Protection, Mobile Code Program, PI Meeting*, Melbourne, Florida; January 2002.

- †*DARPA Organically Assured and Survivable Information Systems (OASIS) Program, PI Meeting*, Santa Fe, New Mexico; July 2001.
- †*ONR Critical Infrastructure Protection, Mobile Code Program, PI Meeting*, Arlington, Virginia; July 2001.
- †*Symposium on Research in Mobile Computing Systems*, Zurich, Switzerland; May 2001.
- †*DARPA Organically Assured and Survivable Information Systems (OASIS) Program, PI Meeting*, Norfolk, Virginia; February 2001.
- *University of Washington and Microsoft Research Summer Institute 2000, “Accelerating the Pace of Software Tools Research: Sharing Infrastructure”*, hosted by C. Chambers, D. Notkin, A. Srivastava, and B. Zorn; Seattle, Washington; August 2000.
- †*DARPA Intrusion Tolerant Systems (ITS) Program, PI Meeting*, Honolulu, Hawaii; July 2000.
- ‡*17th Gesellschaft für Informatik (GI) Workshop on Programming Languages and Computing Concepts (with Special Emphasis on Software Components)*, Bad Honnef, Germany; May 2000.
- †*DARPA Intrusion Tolerant Systems (ITS) Program, PI Meeting*, Aspen, Colorado; February 2000.
- †*DARPA Intrusion Tolerant Systems (ITS) Program, PI Meeting*, Phoenix, Arizona; August 1999.
- *National Science Foundation CAREER Program, PI Meeting*, Washington, D.C.; January 1999.
- †*Southern California Parallel Processing and Computer Architecture Workshop*, Irvine, California; March 1998.
- †*International Workshop on Component-Oriented Programming*, Linz, Austria; July 1996.
- *Third International Workshop on Workstation Operating Systems*, Key Biscayne, Florida; April 1992.

Administrative Service

- Donald Bren School of Information and Computer Science, *Faculty Search Committee, Positions in Systems*, 2017–18, 2018–19.
- Donald Bren School of Information and Computer Science, Chair, *Computing and Network Policy Committee*, 2016–17, 2017–18, 2018–19.
- University of California, Vice Chair, *Irvine Campus Council on Planning and Budget*, 2015–2016, 2016–2017.
- University of California, Member, *Irvine Campus Council on Planning and Budget*, 2014–2015.
- Department of Computer Science, *Software Engineering Steering Committee*, 2013–2014, 2014–2015, 2015–2016, 2016–2017, 2017–2018.
- Department of Computer Science, *CS Graduate Admissions Committee*, 2016–2017.
- Department of Computer Science, *CS Admission and Graduate Student Planning Committee*, 2013–2014, 2014–2015, 2015–2016.
- University of California, Irvine, *5-year Organized Research Unit Review Committee for the Center for Embedded Computer Systems (CECS)*, 2012.
- Donald Bren School of Information and Computer Science, *Executive Committee*, 2007–2008.
- Donald Bren School of Information and Computer Science, Chair, *Computing and Network Policy Committee*, 2005–2006, 2006–2007, 2008–2009.
- Donald Bren School of Information and Computer Science, *Marketing and Outreach Committee*, 2004–2005.

- University of California, *Irvine Campus Council on Undergraduate Admissions and Relations with Schools and Colleges*, 2000–2004.
- Donald Bren School of Information and Computer Science, Chair, *Faculty Search Committee, Position in Security and Cryptography*, 2002–2003.
- Donald Bren School of Information and Computer Science, *Committee on Graduate Policy*, 2002–2003.
- ICS Department, Chair, *Committee on Space Policy*, 2001–2002.
- ICS Department, *Faculty Search Committee, Position in Cryptography and Security*, 2000–2001.
- ICS Department, *Committee on Graduate Policy*, 2000–2001.
- ICS Department, *Ad-Hoc Faculty Search Committee, “Systems” Position*, 1999–2000.
- ICS Department, *Committee on Educational Policy*, 1999–2000.
- ICS Department, *Executive Committee*, 1998–1999.
- ICS Department, *Committee on Undergraduate Policy*, 1998–1999.
- ICS Department, *Faculty Search Committee, Multiple Positions in Interdisciplinary Applications of Computer Science*, 1998–1999. (Committee reviewed 170 applications = 4 linear feet of files and filled three open faculty positions.)
- University of California, *Irvine Campus Committee on Undergraduate Admissions and Relations with Schools and Colleges*, 1997–2000.
- ICS Department, *Committee on Graduate Policy*, 1997–1998.
- ICS Department, *Committee on Graduate Admissions*, 1997–1998.
- ICS Department, *Faculty Search Committee, Position in “Informatics,”* 1997–1998.
- University of California, *Irvine Campus Representative Assembly*, 1996–1997.
- ICS Department, *Committee on Personnel*, 1996–1997.
- ICS Department, *Faculty Search Committee, Position in Software Engineering*, 1995–1996.

Teaching Activities

Teaching Awards

- *Dean's Award for Graduate Student Mentoring*, Donald Bren School of Information and Computer Sciences, UC Irvine, 2016, "For his outstanding mentoring of doctoral students over the last decade."
- *Dean's Award for Graduate Student Mentoring*, Donald Bren School of Information and Computer Sciences, UC Irvine, 2007.
- *Outstanding Professor of the Year Award*, Graduating Class of 2007, UC Irvine.

Post-Doctoral Habilitation Theses Supervised

- Dr. Christian Herrman, Universität Ulm, Germany; thesis: "Verbesserte prozedurale Programmiersprachen" (Improved Procedural Programming Languages); March 2007.

Post-Doctoral Fellows Supervised

1. Dr. Wolfram Amme
(January–December 2000; first subsequent position: Privatdozent at the *University of Jena*, Germany).
2. Dr. Won-Kee Hong
(October 2001–October 2002; first subsequent position: Assistant Professor at *Daegu University*, South Korea).
3. Dr. Fermin Reig
(October 2001–July 2003; first subsequent position: Postdoc at *University of Nottingham*, United Kingdom).
4. Dr. Roxana Diaconescu
(January 2003–September 2004; first subsequent position: PostDoc at *California Institute of Technology (Caltech)*, Pasadena, California).
5. Dr. Christian Probst
(January 2003–May 2005; first subsequent position: Assistant Professor at the *Technical University of Denmark (DTU)*, Lyngby, Denmark).
6. Dr. Andreas Gal
(January 2007–February 2010, first subsequent position: Researcher at *Mozilla*, Mountain View, California).
7. Dr. Christian Stork
(March 2007–September 2008).
8. Dr. Christian Wimmer
(July 2008–April 2011, first subsequent position: Principal Member of Technical Staff, *Oracle Sun Labs*, Redwood Shores, California).
9. Dr. Stefan Brunthaler
(April 2011–June 2015), first subsequent position: Key Researcher at *SBA Research*, Vienna, Austria).
10. Dr. Per Larsen
(September 2011–June 2015, first subsequent position: Chief Executive Officer of *Immunant*, Irvine, California).
11. Dr. Stijn Volckaert
(December 2015–July 2018, first subsequent position: Assistant Professor at *KU Leuven*, Belgium).
12. Dr. Yeoul Na
(since July 2016).

13. Dr. David Gens
(since March 2019).
14. Dr. Adrian Dabrowski
(since May 2019).

Graduated Ph.D. Students (Principal Advisor and Dissertation Committee Chair)

1. Thomas Kistler
(affiliated in April 1995, candidacy: February 1998, final defense: November 1999; thesis: “Continuous Program Optimization;” first employment after graduation: *Transmeta, Inc.*, Santa Clara, California).
2. Peter H. Fröhlich
(affiliated in September 1998; advanced to candidacy in May 2001; final defense in March 2003; thesis: “The Structure of Component-Oriented Programming Languages;” first employment after graduation: *University of California, Riverside*, California).
3. Jeffery von Ronne
(affiliated in September 1999; advanced to candidacy in February 2003; final defense in July 2005; thesis: “A Safe and Efficient Machine-Independent Code Transportation Format Based on Static Single Assignment Form and Applied to Just-In-Time Compilation;” first employment after graduation: *University of Texas at San Antonio*).
4. Vivek Haldar
(affiliated in August 2000; advanced to candidacy: November 2002; final defense: February 2006; thesis: “Semantic Remote Attestation;” first employment after graduation: *Google*, Santa Monica, California).
5. Efe Yardimci
(affiliated in August 2001; advanced to candidacy: November 2003; final defense: March 2006; thesis: “Exploiting Parallelism to Improve the Performance of Sequential Binary Executables;” first employment after graduation: *Advanced Micro Devices (AMD)*, Santa Clara, California).
6. Christian H. Stork
(affiliated in September 1998; advanced to candidacy: May 2001; final defense: August 2006; thesis: “WELL: A Language-Agnostic Foundation for Compact and Provably Safe Mobile Code;” first employment after graduation: Postdoc at *University of California, Irvine*).
7. Deepak Chandra
(affiliated in August 2001; advanced to candidacy: March 2004; final defense: September 2006; thesis: “Information Flow Analysis and Enforcement in Java Bytecode;” first employment after graduation: *Google*, Irvine, California).
8. Andreas Gal
(affiliated in January 2002; advanced to candidacy: December 2003; final defense: November 2006; thesis: “Efficient Bytecode Compilation and Verification in a Virtual Machine;” first employment after graduation: Postdoc at *University of California, Irvine*).
9. Matthew Beers
(affiliated in September 1999; advanced to candidacy: July 2002; final defense: March 2007; thesis: “Shifting the Burden of Code Optimization to the Code Producer;” first employment after graduation: *Ocean Tomo* Intellectual Capital Equity, San Francisco, California).
10. Ning Wang
(affiliated in September 2001; advanced to candidacy: September 2004; final defense: May 2007; thesis: “From Assumptions to Assertions: A Sound and Precise Points-to Analysis for the C Language;” first employment after graduation: *Fortify Software*, Palo Alto, California).

11. Vasanth Venkatachalam
(affiliated in September 2002; advanced to candidacy: September 2003; final defense: May 2007; thesis: “Self-Calibrating Processor Speed: A New Feedback Loop For Dynamic Voltage Scaling Control;” first employment after graduation: *Advanced Micro Devices (AMD)*, Austin, Texas).
12. Lei Wang
(affiliated in June 2001; advanced to candidacy: September 2004; final defense: June 2009; thesis: “Automatic Program Partitioning to Alleviate Resource Constraints of Object-Oriented Applications;” first employment after graduation: *Microsoft*, Redmond, Washington).
13. Babak Salamat
(affiliated in January 2007; advanced to candidacy: May 2007; final defense: June 2009; thesis: “Multi-Variant Execution: Run-Time Defense Against Malicious Code Injection Attacks;” first employment after graduation: *Yahoo*, Sunnyvale, California).
14. Michael Bebenita
(affiliated in January 2007; advanced to candidacy in May 2009; final defense: October 2011; thesis: “Trace-Based Compilation and Optimization in Meta-Circular Virtual Execution Environments;” first employment after graduation: *Mozilla*, Mountain View, California).
15. Gregor Wagner
(affiliated in September 2007; advanced to candidacy in May 2009; final defense: October 2011; thesis: “Domain Specific Memory Management in a Modern Web Browser;” first employment after graduation: *Mozilla*, Mountain View, California).
16. Mason Liu Chang
(affiliated in June 2007; advanced to candidacy in May 2009; final defense: February 2012; thesis: “Efficient Analysis and Optimization of Dynamically Typed Languages;” first employment after graduation: *Mozilla*, Mountain View, California).
17. Todd Morris Jackson
(affiliated in September 2007; advancement to candidacy in June 2009; final defense: May 2012; thesis: “On the Design, Implications, and Effects of Implementing Software Diversity for Security;” first employment after graduation: *Google*, Mountain View, California).
18. Christoph Kerschbaumer
(affiliated in Summer 2010; advancement to candidacy in November 2011; final defense: March 2014; thesis: “Probabilistic Information Flow Control in Modern Web Browsers;” first employment after graduation: *Mozilla*, Mountain View, California).
19. Eric Hennigan
(affiliated in July 2008; advancement to candidacy in April 2011; final defense: December 2014; thesis: “From FlowCore to JitFlow: Improving the Speed of Information Flow in JavaScript;” first employment after graduation: *Google*, Mountain View, California).
20. Marcelo Cintra
(affiliated in December 2007; advancement to candidacy in November 2009, final defense: April 2015; thesis: “Just-in-Time Compilation Techniques for Hardware/Software Co-Designed Processors;” first employment after graduation: *Intel*, Santa Clara, California).
21. Andrei Homescu
(affiliated in Fall 2010; advancement to candidacy in March 2012, final defense: April 2015; thesis: “Securing Statically and Dynamically Compiled Programs using Software Diversity;” first employment after graduation: *Immunant*, Irvine, California).
22. Codrut Stancu
(affiliated in Summer 2012; advancement to candidacy in May 2013, final defense: May 2015; thesis: “Safe and

Efficient Hybrid Memory Management for Java;” first employment after graduation: *Oracle*, Redwood Shores, California).

23. Wei Zhang
(affiliated in Spring 2011; advancement to candidacy in November 2011, final defense: June 2015; thesis: “Efficient Hosted Interpreters for Dynamic Languages;” first employment after graduation: *Twitter*, San Francisco, California).
24. Stephen Crane
(affiliated in Fall 2011; advancement to candidacy in August 2013, final defense: June 2015; thesis: “Enhancing and Extending Software Diversity;” first employment after graduation: *Immunant*, Irvine, California).
25. Gulfem Savrun Yeniceri
(affiliated in Fall 2010; advancement to candidacy in January 2013, final defense: November 2015; thesis: “Efficient Interpreters and Profilers for Hosted Dynamic Languages;” first employment after graduation: *Intel*, Santa Clara, California).
26. Julian Lettner
(affiliated in Fall 2013; advancement to candidacy in March 2016, final defense: August 2018; thesis: “Finding and Mitigating Memory Corruption Errors in Systems Software;” first employment after graduation: *Apple*, Cupertino, California).
27. Brian Belleville
(affiliated in Fall 2013; advancement to candidacy in June 2016, final defense: August 2018; thesis: “Security Applications of Static Program Analysis;” first employment after graduation: *Google*, Mountain View, California).
28. Mohaned Qunaibit
(since Summer 2014; advancement to candidacy in March 2016, final defense: March 2019; thesis: “Accelerating Dynamically-Typed Language on Heterogeneous Platforms”).

Graduate Students Supervised as Principal Academic Advisor and Committee Chair

Advanced to Ph.D. Candidacy (in order of advancement date)

1. Joseph Nash (since Summer 2015; advancement to candidacy in May 2018)
2. Taemin Park (since Summer 2015; advancement to candidacy in May 2018)
3. Prabhu Karthikeyan Rajasekaran (since Spring 2015; advancement to candidacy in May 2018)
4. Alexios Voulimeneas (since Summer 2015; advancement to candidacy in June 2018)
5. Anil Altinay (since Summer 2015; advancement to candidacy in June 2018)
6. Paul Kirth (since Fall 2016; advancement to candidacy in March 2019)
7. Dokyung Song (from Fall 2016; advancement to candidacy in May 2019)

Not Yet Advanced to Candidacy (in order of affiliation date)

8. Fabian Parzefall (from Summer 2018)
9. Mitchel Dickerson (from Summer 2018)
10. Matthew Dees (from Summer 2018)
11. Min-Yi Hsu (from Fall 2018)

Graduated Ph.D. Students (Co-Advisor and “Opponent” During Final Dissertation Defense)

1. Christian Wimmer, University of Linz, Austria
(final defense: March 2008; thesis: “Automatic Object Inlining in a Java Virtual Machine”).
2. Stefan Brunthaler, Technical University of Vienna, Austria
(final defense: February 2011; thesis: “Purely Interpretative Optimizations”).
3. Thomas Würthinger, Johannes-Kepler University of Linz, Austria
(final defense: April 2011; thesis: “Dynamic Code Evolution for Java”).
4. Christian Häubl, Johannes-Kepler University of Linz, Austria
(final defense: February 2015; thesis: “Generalized Trace Compilation for Java”).
5. Stijn Volckaert, University of Ghent, Belgium
(final defense: October 2015; thesis: “Advanced Techniques for Multi-Variant Execution”).

Other Ph.D. Students

Dissertation Committee Member

- Byron Hawkins, UC Irvine
(final defense: August 2017; committee chair: Brian Demsky; thesis: “Introspective Intrusion Detection”).
- Andreas Gerstlauer, UC Irvine
(final defense: April 2004, committee chair: Daniel D. Gajski; thesis: “Modeling Flow for Automated System Design and Exploration”).
- Ana Lucia Velloso Azevedo, UC Irvine
(final defense: October 2002; committee chair: Alexandru Nicolau; thesis: “Annotation-based Compiler Technology”).
- Chang Liu, UC Irvine
(final defense: August 2002, committee chair: Debra J. Richardson; thesis: “Redundant Arrays of Independent Components”).
- Martin Burtscher, University of Colorado at Boulder
(final defense: April 2000, committee chair: Benjamin Zorn; thesis: “Improving Context-Based Load Value Prediction”).
- Jianwen Zhu, UC Irvine
(final defense: September 1999; committee chair: Daniel D. Gajski; thesis: “Behavioral Synthesis from an Extensible Object Oriented Language”).

Candidacy Committee Member)

- Tyler Kaczmarek, UC Irvine
(candidacy: December 2015; committee chair: Gene Tsudik).
- Lu Fang, UC Irvine
(candidacy: May 2014; committee chair: Guoqing Xu).
- Nicolae Savoiiu, UC Irvine
(candidacy: September 1999; committee chair: Alexandru Nicolau).

M.Sc. Students Graduated from UC Irvine with Thesis Option

- Wail Alkowaileet, M.S. thesis committee member, graduated October 2013 (thesis: “NUMA-aware multicore Matrix Multiplication;” committee chair: Isaac Scherson).
- Alexander Yermolovich, primary M.S. advisor / committee chair, completed M.S. degree in May 2009 (thesis: “Efficient Execution of Binary and Guest Virtual Machines on Platform Independent Host Virtual Machines”).
- Mason Liu Chang, primary M.S. advisor / committee chair, completed M.S. degree in May 2009 (thesis: “Tracing for Web 3.0 – Trace Compilation for the Next Generation Web Applications”).
- Songmei Han, primary M.S. advisor, graduated with a M.S. in Computer Science in June 2003 (she also received a Ph.D. in Cognitive Science, for which Barbara Doshier was the advisor); subsequently a tenure-track Assistant Professor of Cognitive Science and Computer Science at SUNY Oswego and now Usability Engineer at Apollo Group.
- Anjum Gupta, M.S. thesis committee member, graduated June 2003 (thesis: Design and Implementation of an Adaptive Cache on a Configurable Processor; committee chair: Rajesh Gupta).

Other Graduate Advising

- Dixin Zhou, primary M.S. advisor, Spring 2018 – Spring 2019, graduated June 2019.
- Faraz Zaerpoor, primary M.S. advisor, Fall 2016 – Spring 2018.
- Anton Vasick, primary M.S. advisor, Summer 2015 – Spring 2018.
- Mark Murphy, research advisor, Fall 2010 – Fall 2015.
- Divya Varshini Agavalam Padmanabhan, primary M.S. advisor, graduated June 2016.
- Nikhil Gupta, primary M.S. advisor, graduated June 2016.
- Roeland Singer-Heinze, primary M.S. advisor, graduated June 2016.
- Stephen Neisius, primary M.S. advisor, graduated Summer 2014.
- Karthikeyan Manivannan, primary advisor, 2007– 2011.
- Sergiy Zhenochin, primary M.S. advisor / committee chair, graduated Fall 2001.
- Prashant Saraswat, primary M.S. advisor / committee chair, graduated Fall 2001.
- Hans-Christian Stadler, primary M.S. advisor / committee chair, graduated June 1998.

Undergraduate Honors Students Graduated from UC Irvine

- Eric Thomas Parsons (summer research advisor, Summer 2018).
- Muneeb Baig (honors research advisor); graduated Magna Cum Laude and Phi Beta Kappa in 2007; honors thesis: “Optimizing Array Bound Checking During Trace-Based Compilation”.
- Michael Masukawa (honors research advisor); graduated Summa Cum Laude and Phi Beta Kappa in 2007; honors thesis: “Dynamic Taint Propagation in Java Web Applications”.
- Jesse Morrow (honors research advisor); graduated Magna Cum Laude and Phi Beta Kappa in 2005.
- Matthew Chu (honors research advisor); graduated Phi Beta Kappa in 2004.
- Zachary Mouri (honors research advisor); graduated Phi Beta Kappa in 2004.
- Ronald Harvest (honors research advisor); graduated Summa Cum Laude and Phi Beta Kappa in 1999.
- Calvin Shen (honors research advisor); graduated Cum Laude in 1999.

Other Undergraduate Advising

- Rasmus Tjalk-Boggild, visiting from DTU Lynby, Denmark, faculty research advisor; Summer 2016.
- Thomas Bourgenolle, visiting from ENSTA ParisTech, Paris, France, faculty research advisor; Summer 2015.
- Martin Imre, visiting from Technical University of Vienna, Austria, faculty research advisor; Summer 2015.
- Dominik Infuehr, visiting from Technical University of Vienna, Austria, faculty research advisor; Summer 2015.
- David Poetzsch-Heffter, visiting from University of Kaiserslautern, Germany, faculty research advisor; Summer 2015.
- Aditiya Verma, visiting from IIT (BHU) Varanasi, India, faculty research advisor; Summer 2015.
- William Lee, Troy Tech Senior Internship, supervisor; Summer 2015.
- Christos Ioannidis, visiting from University of Thessaly, Greece, faculty research advisor; Summer 2014.
- Mohit Mishra, visiting from Indian Institute of Technology, Varanasi, faculty research advisor; Summer 2014.
- Michalis Papamichail, visiting from Aristotle University of Thessaloniki, Greece, faculty research advisor; Summer 2014.
- Martin Schleiss, visiting from Technical University of Vienna, Austria, faculty research advisor; Summer 2014.
- Henry Elias Hernandez, faculty research advisor, Summer 2014.
- Daniel Nima Salehi, faculty research advisor, Summer 2014.
- Michael Stewart, faculty research advisor, Summer/Fall 2011.
- Jeffrey Bosboom, NSF Research Experiences for Undergraduates Summer Scholar, faculty advisor, 2011.
- Shawn Merrill, NSF Research Experiences for Undergraduates Summer Scholar, faculty advisor, 2011.
- Chris Austin, NSF Research Experiences for Undergraduates Summer Scholar, faculty advisor, 2010.
- Daniel A. Ehrenberg, Carleton University, NSF Research Experiences for Undergraduates Summer Scholar, faculty advisor, 2010.
- Sean Kocol, honors research advisor, 2009.
- Jonathan Mood, honors research advisor, 2009.
- Adrian Tran, honors research advisor, 2009.
- Yaoxiang Zhou, honors research advisor, 2008.
- Raymond Yu, honors research advisor, 2007/08.
- Stephen C. Reed, California Alliance for Minority Participation in Science (CAMP) Summer Scholar, faculty advisor, 2004.

Visiting Diploma Students Supervised at UC Irvine

- Urs Fässler, ETH Zürich, Switzerland, co-supervised with Th. Gross; March–September 2012.
- Alen Stojanov, École polytechnique fédérale de Lausanne (EPFL), Switzerland, co-supervised with M. Odersky; September 2011–March 2012.
- Dominik Lichtenauer, Johannes-Kepler University of Linz, Austria, co-supervised with H. Mössenböck, September 2011–March 2012.
- Stefan Rath, Technische Universität Graz, Austria, co-supervised with Ch. Steger; July–December 2010.
- Franz Maier, Technische Universität Graz, Austria, co-supervised with Ch. Steger; March–September 2010.
- Christoph Kerschbaumer, Technische Universität Graz, Austria, co-supervised with Ch. Steger; March–September 2008.
- Giacomo Amorosa, ETH Zürich, Switzerland, co-supervised with J. Gutknecht; February–August 2008.
- Katharina Seke, Technische Universität Graz, Austria, co-supervised with Ch. Steger; July 2006–October 2006.
- Gregor Wagner, Technische Universität Graz, Austria, co-supervised with Ch. Steger; March 2006–October 2006.
- Albert Noll, Technische Universität Graz, Austria, co-supervised with Ch. Steger; March 2006–September 2006.
- Michael Rauch, Technische Universität Graz, Austria, co-supervised with Ch. Steger; March 2006–September 2006.
- Isabella Thomm, Universität Erlangen-Nürnberg, Germany, co-supervised with W. Schroeder-Preikschat; November 2005–February 2006.
- Michael Stalkerich, Universität Erlangen-Nürnberg, Germany, co-supervised with W. Schroeder-Preikschat; December 2004–February 2005 and November 2005–February 2006.
- Nicolas Marochow, Fachhochschule Braunschweig-Wolfenbüttel, Germany, co-supervised with R. Rüdiger; September 2004–January 2005.
- Jan Peterson, Universität Jena, Germany, co-supervised with W. Amme; May–September 2004.
- Tobias Körner, Fachhochschule Braunschweig-Wolfenbüttel, Germany, co-supervised with R. Rüdiger; March–August 2003.
- Alexander Apel, Universität Jena, Germany, co-supervised with W. Amme; September–November 2003.
- Christian Rattei, Fachhochschule München, Germany, co-supervised with K. Köhler; April–November 2000.
- Joachim Büchse, ETH Zürich, Switzerland, co-supervised with J. Gutknecht, 1998.
- M. Burtscher, ETH Zürich, 1996.
(work conducted at Irvine but thesis submitted in Zurich while Franz still had a formal association with ETH)
- M. Dätwyler, ETH Zürich, 1996.
(work conducted at Irvine but thesis submitted in Zurich while Franz still had a formal association with ETH)

Diploma Students Supervised at ETH Zurich († = co-supervised with N. Wirth)

- E. Brandenberger, Oberon Module Interchange auf Intel-Prozessoren, 1996.
- D. Posva, Dynamische Reoptimierung auf einem RISC, 1996.
- M. Sperisen, Executable Content in WWW-Dokumenten: Java, 1996.
- †H. Buchser, Portable Objektfiles und codegenerierender Lader, 1995.
- †H. Domjan, Metaprogrammierung, 1995.

- †O. Dreer, “Slim Binaries” auf Macintosh, 1995.
- †Th. Kistler, Smartest Recompilation, 1995.
- †Ch. Denzler, A Message Mechanism for Oberon, 1993.
- †E. Oertli, Oberon-2 für Macintosh, 1993.
- †I. Posva, Elimination redundanter Tests durch Programmanalyse, 1993.
- †Th. Bühlmann, Call Optimization for the MacOberon Compiler, 1992.
- †S. Ludwig, A Portable Object and Symbol File Format for Oberon, 1991.
- †S. Meier, Zeichenerkennung mittels Strukturanalyse, 1990.

Compact Courses and Tutorial Presentations at Professional Meetings

- M. Franz; *Language-Based Security*; half-day tutorial presented at the 2006 Joint Modular Languages Conference (JMLC), Oxford, United Kingdom; September 2006.
- M. Franz; *Safe Code: Language-Based Security in a Networked World*; one-day tutorial presented to the Orange County Chapter of the Institute of Electrical and Electronics Engineers (IEEE); Costa Mesa, California; July 2004.
- M. Franz; *A Tutorial On Language Based Security*; one-day tutorial presented in the ICS Tutorials Series; UCI University Club, Irvine, California; July 2003.
- M. Franz (supervisor and chief instructor); *Moderne Programmierparadigmen (Modern Programming Paradigms)*; 3-day compact course, Post-College Education Program, ETH Zürich; September 1995.
- M. Franz; *Tutorial on Extensible Programming in Oberon*; European Conference on Object-Oriented Programming (ECOOP '95), Aarhus, Denmark, August 1995.
- M. Franz (supervisor and chief instructor); *Moderne Programmierparadigmen (Modern Programming Paradigms)*; 3-day compact course, Post-College Education Program, ETH Zürich; April 1995.
- M. Franz (supervisor and chief instructor); *Programmieren in Oberon (Programming in Oberon)*; 2-day compact course, Post-College Education Program, ETH Zürich; September 1994.
- M. Franz (supervisor and chief instructor); *Erweiterbare Systeme mit Oberon (Extensible Systems with Oberon)*; 1-day advanced-level compact course, Post-College Education Program, ETH Zürich; September 1994.

APPENDIX B-1



US008407609B2

(12) **United States Patent**
Turner

(10) **Patent No.:** **US 8,407,609 B2**

(45) **Date of Patent:** **Mar. 26, 2013**

(54) **SYSTEM AND METHOD FOR PROVIDING AND TRACKING THE PROVISION OF AUDIO AND VISUAL PRESENTATIONS VIA A COMPUTER NETWORK**

(58) **Field of Classification Search** 715/716, 715/719, 733, 764, 765, 760; 709/203, 204, 709/217, 231

See application file for complete search history.

(75) Inventor: **Tod C. Turner**, Kenmore, WA (US)

(56) **References Cited**

(73) Assignee: **LINQware Inc.**, Kenmore, WA (US)

U.S. PATENT DOCUMENTS

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 559 days.

6,606,102 B1* 8/2003 Odom 715/745
2002/0198781 A1* 12/2002 Cobley 705/14
2006/0224693 A1* 10/2006 Gaidemak et al. 709/217
2011/0082754 A1* 4/2011 Shuster 705/14.68

* cited by examiner

(21) Appl. No.: **12/545,131**

Primary Examiner — Xiomar L Bautista

(22) Filed: **Aug. 21, 2009**

(74) *Attorney, Agent, or Firm* — Cozen O'Connor

(65) **Prior Publication Data**

US 2010/0050096 A1 Feb. 25, 2010

(57) **ABSTRACT**

Related U.S. Application Data

(60) Provisional application No. 61/090,672, filed on Aug. 21, 2008.

A method for tracking digital media presentations: providing a corresponding web page for each digital media presentation to be delivered; providing identifier data to the user's computer; providing a timer applet to the user's computer; and storing data indicative of received identifier data; wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from a first computer system directly to the user's computer independent of the first computer system; and stored data is indicative of an amount of time the digital media presentation data is streamed from the second computer system to the user's computer.

(51) **Int. Cl.**
G06F 3/00 (2006.01)
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **715/760; 715/716; 715/733; 709/231**

3 Claims, 10 Drawing Sheets

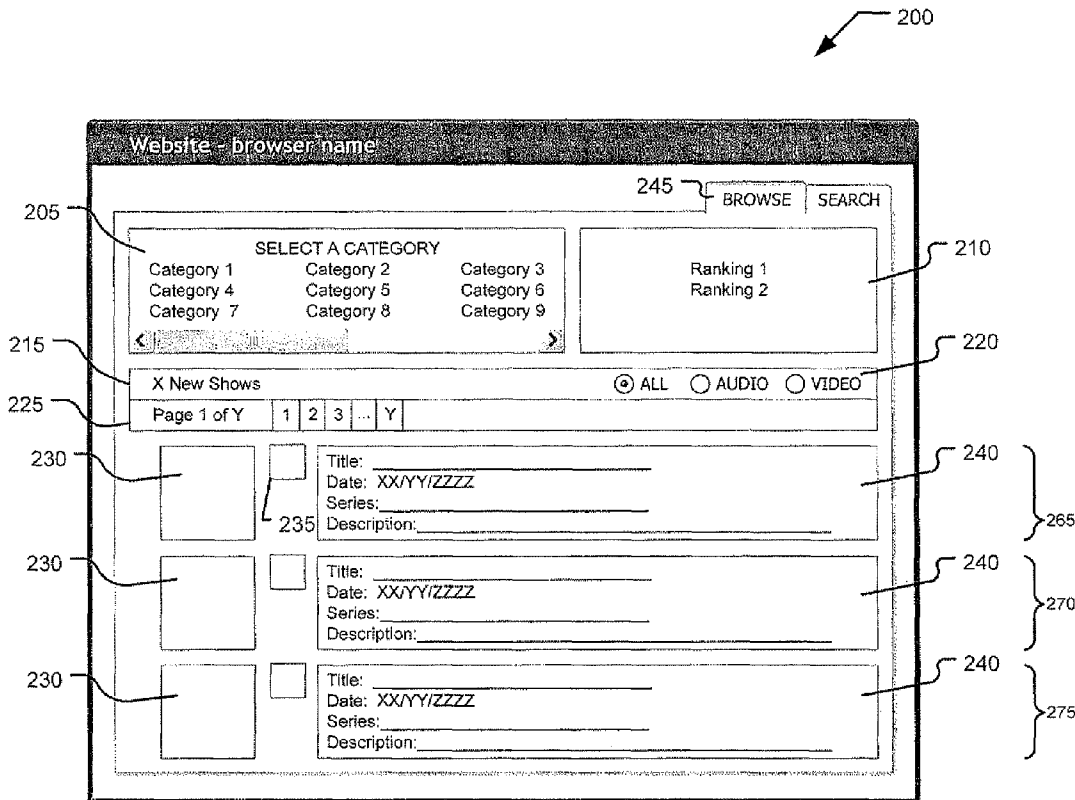


Fig. 1

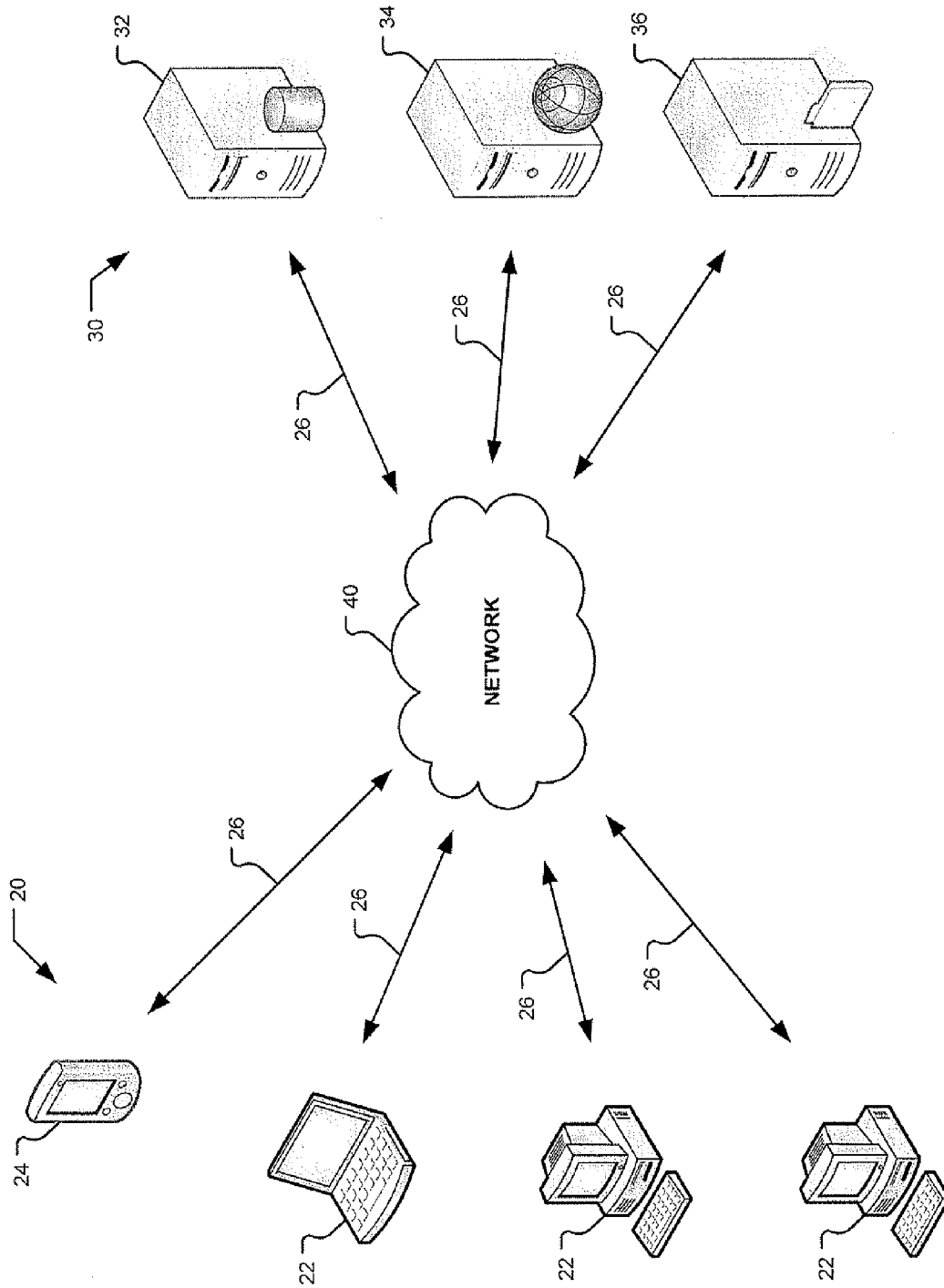


Fig. 2

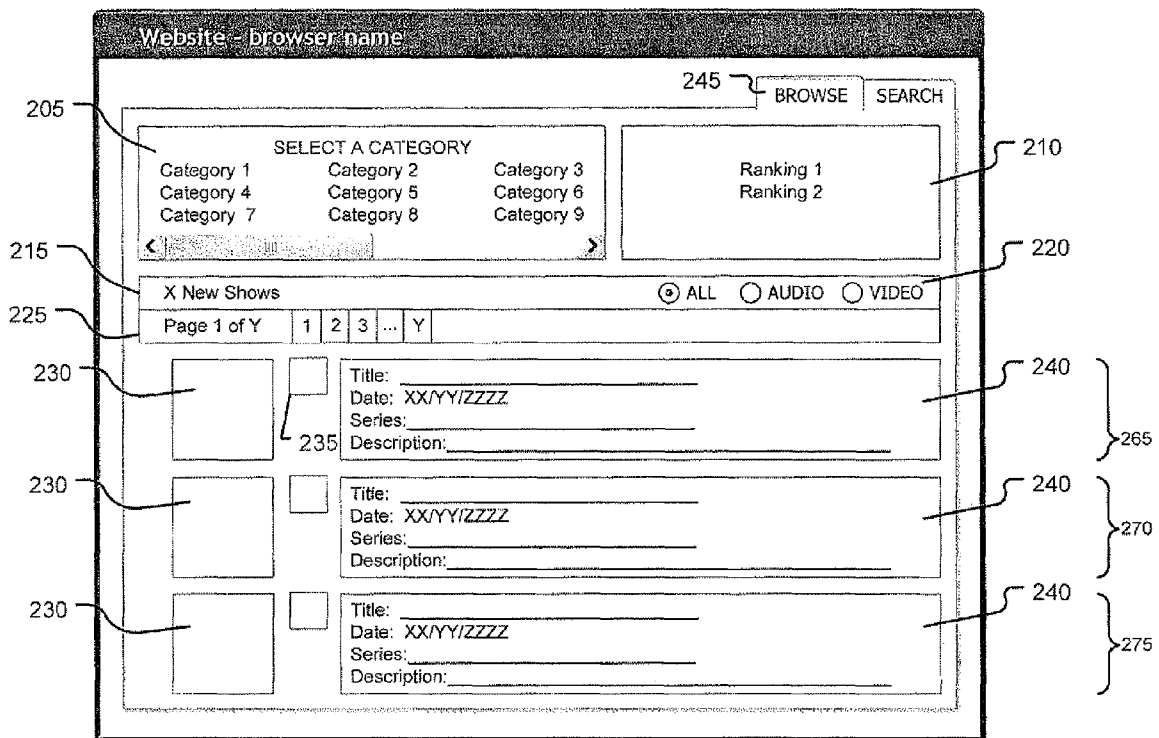
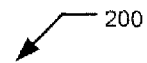


Fig. 3



Website - browser name

BROWSE SEARCH

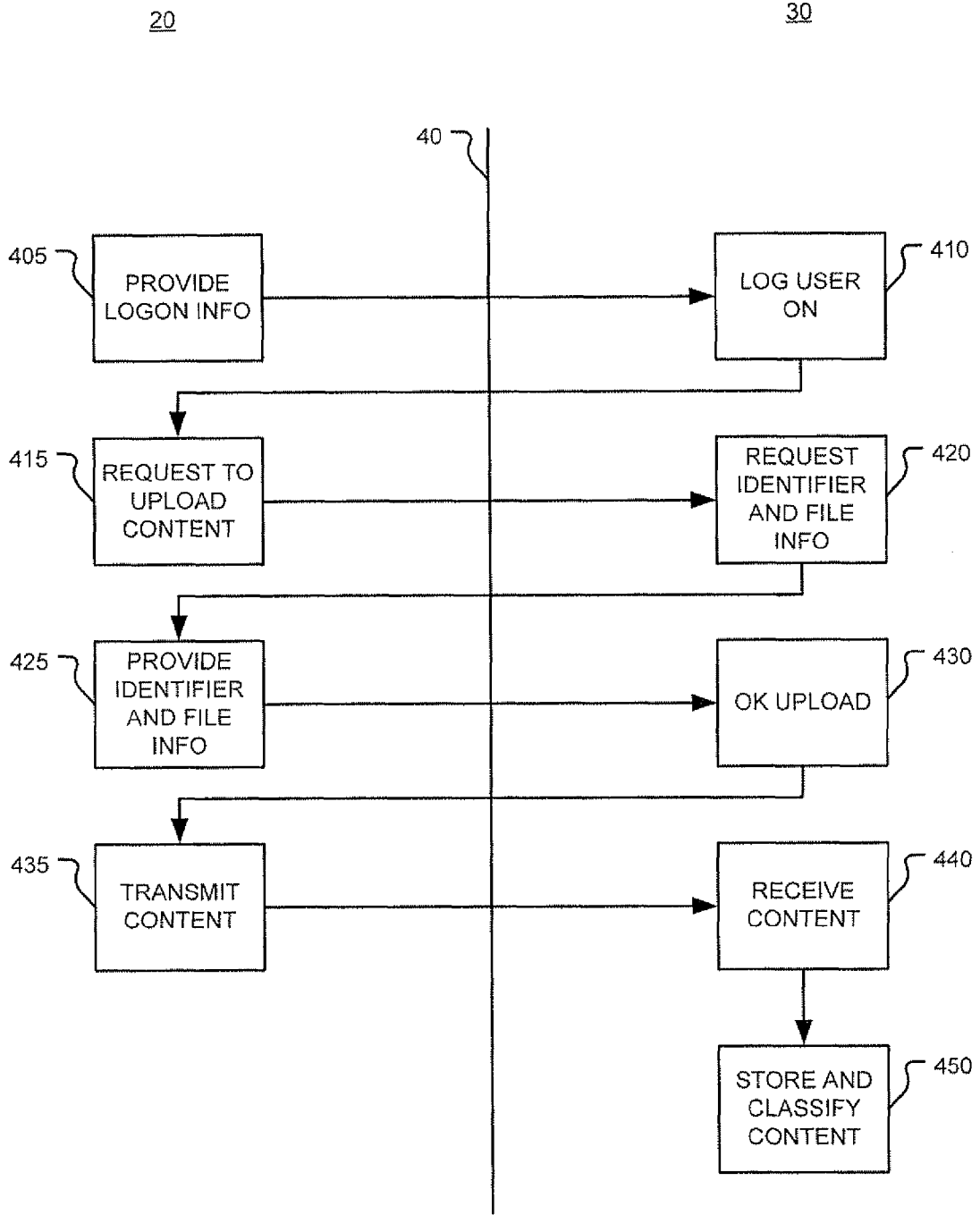
255 { Enter Text 260 { SEARCH 250 {

X New Shows ALL AUDIO VIDEO

Page 1 of Y 1 2 3 ... Y

| | | |
|--------------------------|--------------------------|---|
| <input type="checkbox"/> | <input type="checkbox"/> | Title: _____ Date: XX/YY/ZZZZ Series: _____ Description: _____ |
| <input type="checkbox"/> | <input type="checkbox"/> | Title: _____ Date: XX/YY/ZZZZ Series: _____ Description: _____ |
| <input type="checkbox"/> | <input type="checkbox"/> | Title: _____ Date: XX/YY/ZZZZ Series: _____ Description: _____ |

Fig. 4



400

Fig. 5

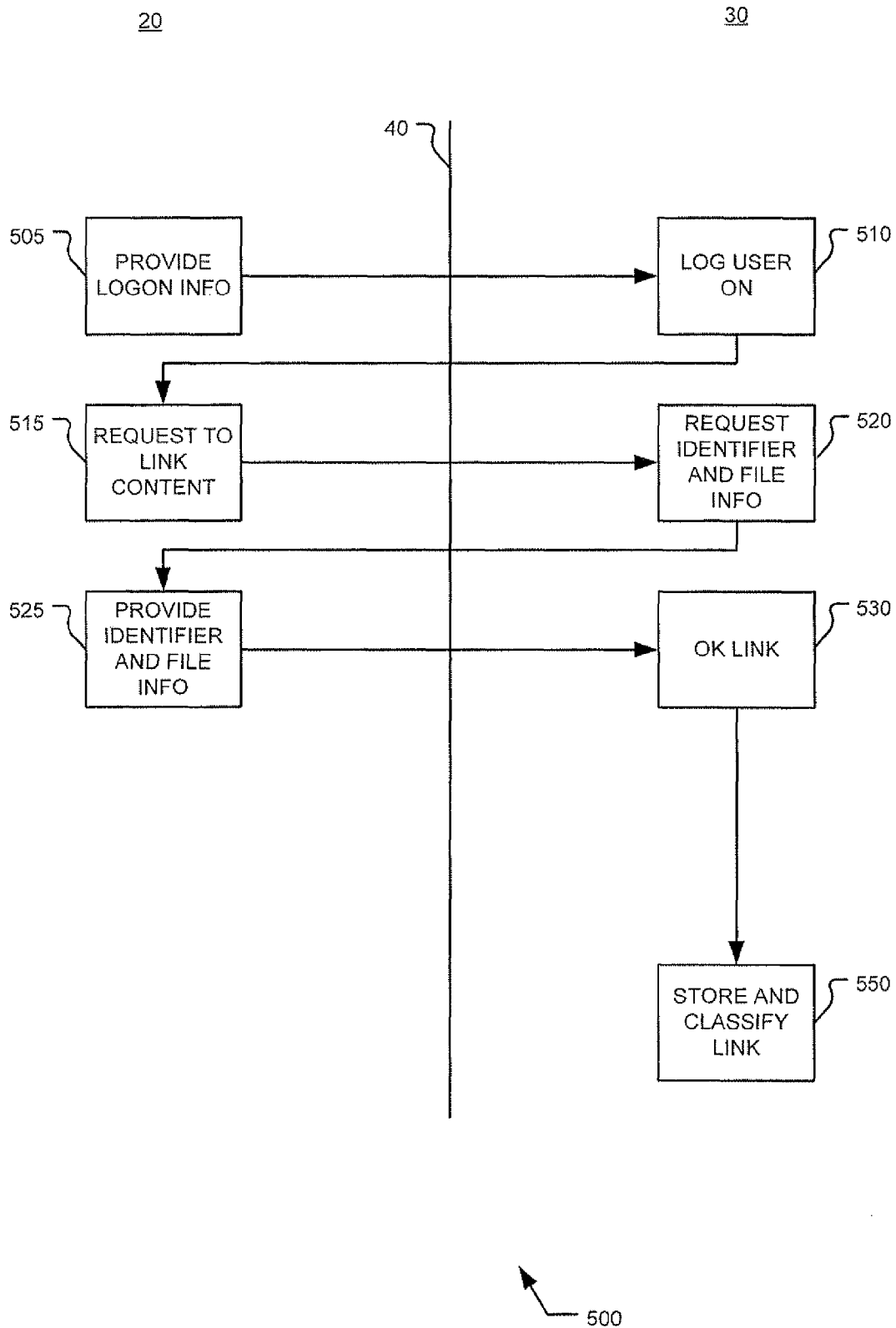
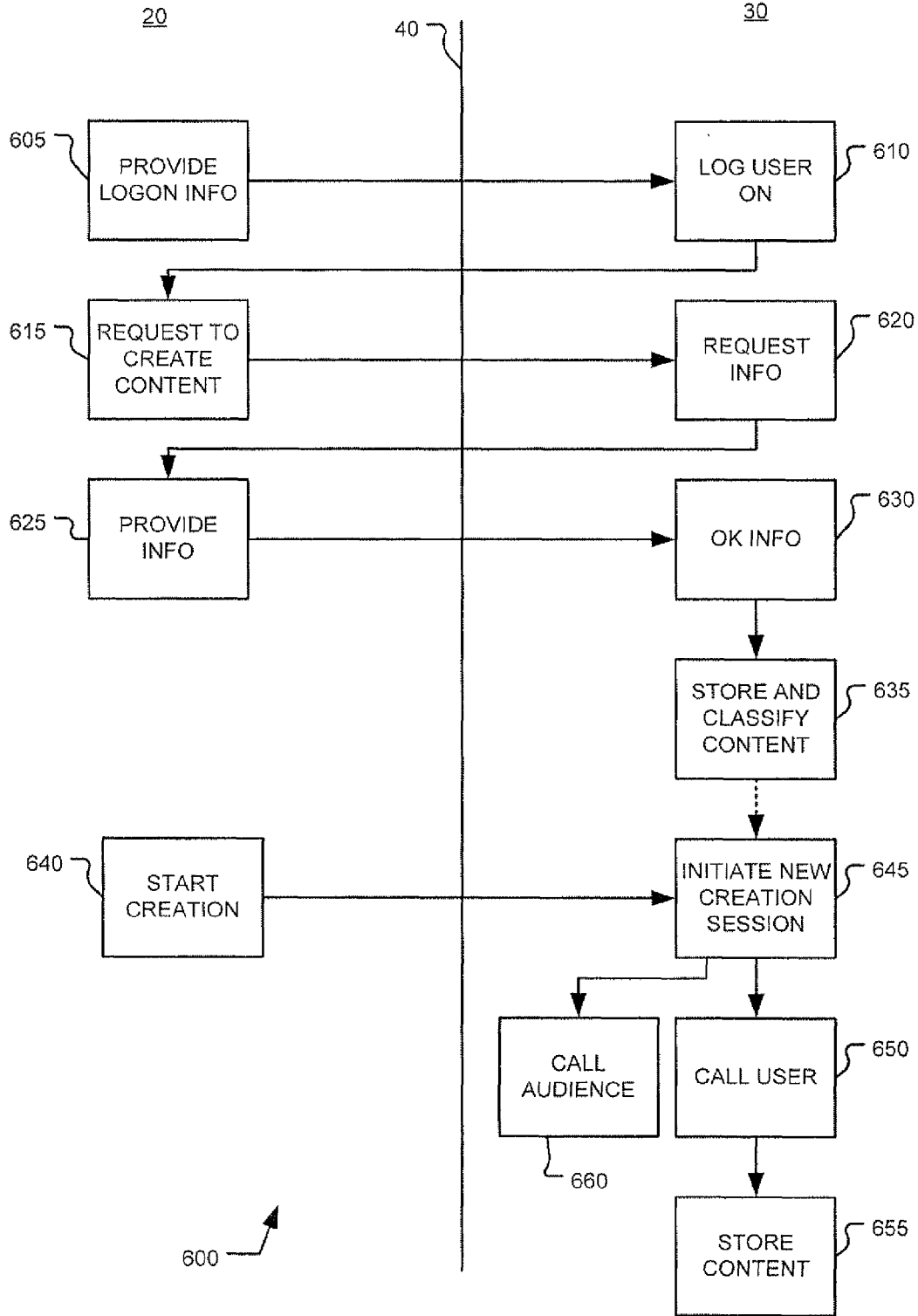


Fig. 6



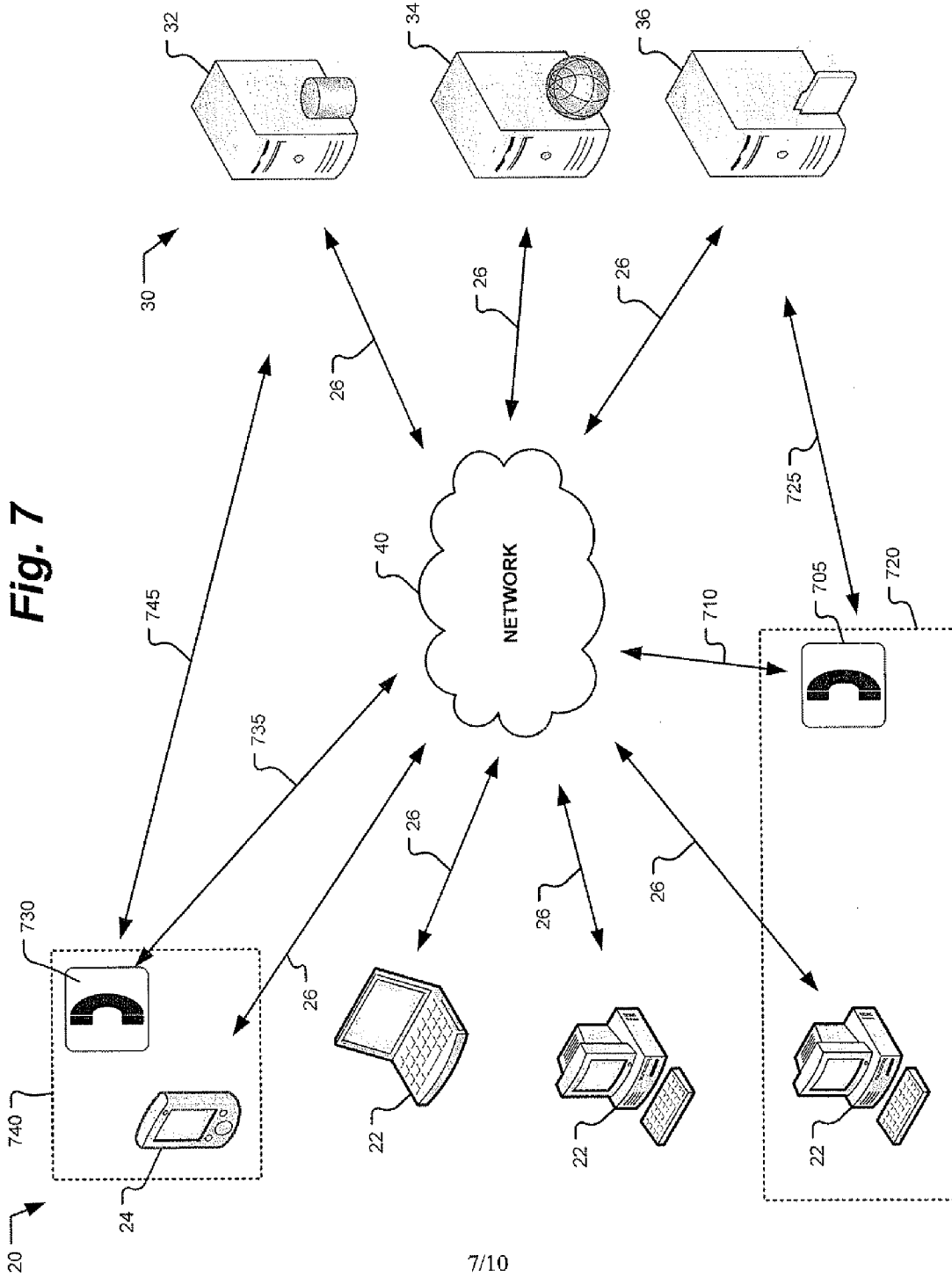


Fig. 8

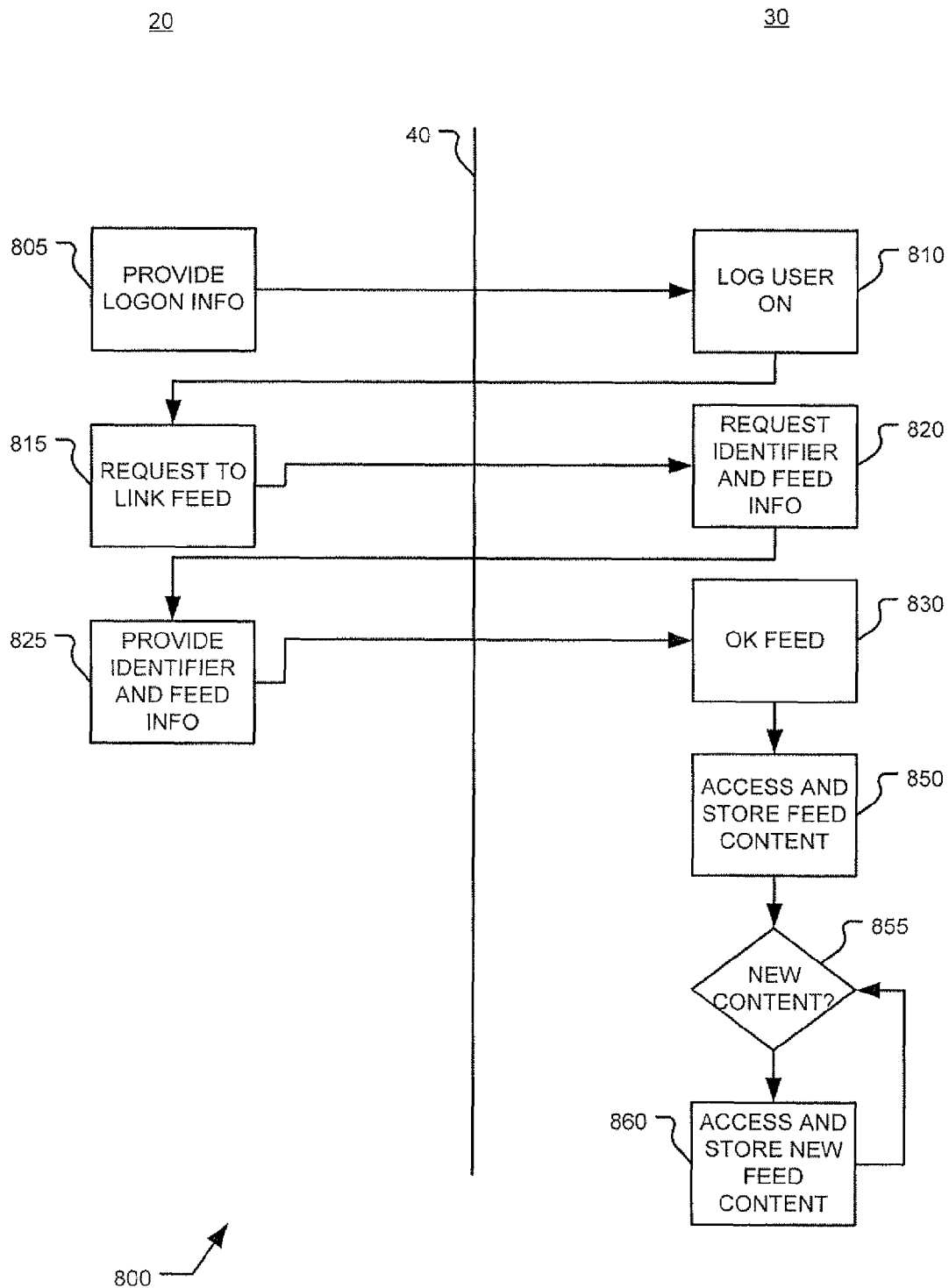


Fig. 9

900

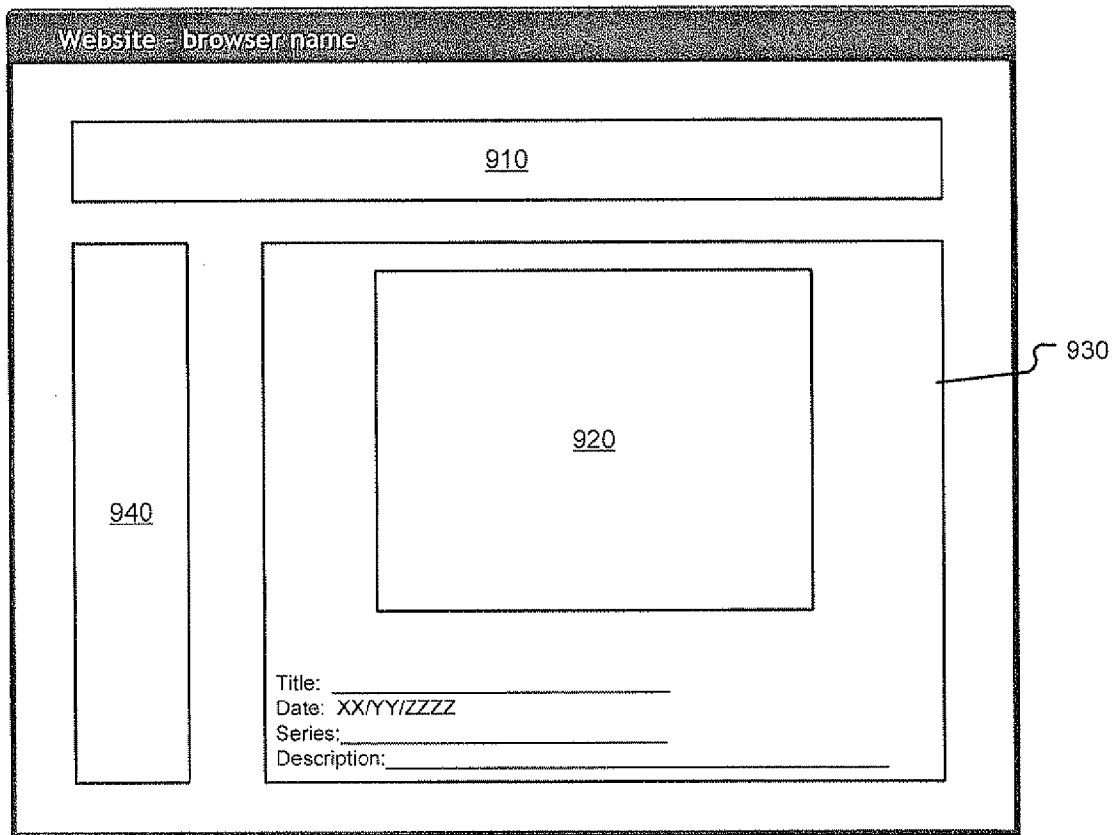
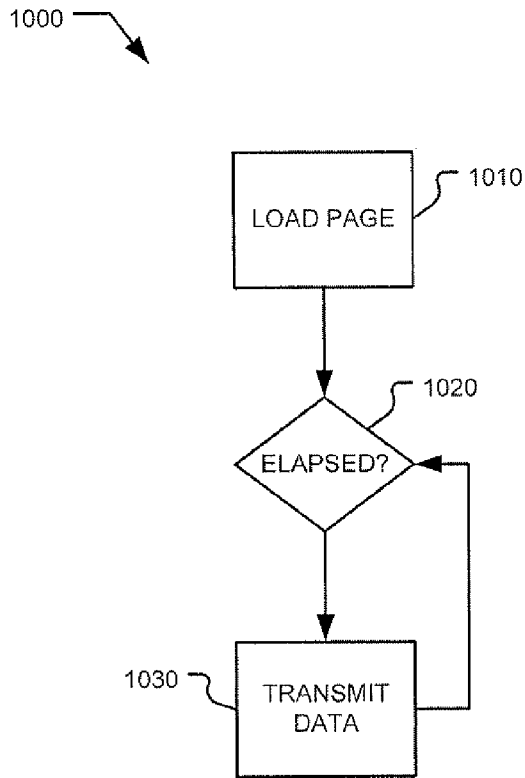


Fig. 10



1

**SYSTEM AND METHOD FOR PROVIDING
AND TRACKING THE PROVISION OF AUDIO
AND VISUAL PRESENTATIONS VIA A
COMPUTER NETWORK**

RELATED APPLICATIONS

This application claims priority of U.S. Patent Application Ser. No. 61/090,673 entitled: CONTENT, TRAFFIC AND ADVERTISING ENGINE, SYSTEM AND METHOD; Ser. No. 61/090,680 entitled: SYSTEM AND METHOD FOR AGGREGATING AND PROVIDING AUDIO AND VISUAL PRESENTATIONS VIA A COMPUTER NETWORK; Ser. No. 61/090,678 entitled: CONTENT, TRAFFIC AND ADVERTISING ENGINE, SYSTEM AND METHOD; Ser. No. 61/090,688 entitled: SYSTEM AND METHOD OF VALIDATING CONTENT, TRAFFIC AND ADVERTISING IN A COMPUTING APPLICATION AND ENGINE; Ser. No. 61/090,681 entitled: DYNAMIC READ THROUGH DATA COLLECTION AND AD DELIVERY SYSTEM AND METHOD OF SAME; Ser. No. 61/090,684 entitled: SYSTEM AND METHOD FOR TRAFFIC IN A CONTENT AND ADVERTISING ENGINE; and Ser. No. 61/090,672 entitled: System and Method for Providing and Tracking the Provision of Audio and Visual Presentations via a Computer Network, all having common inventor Tod C. Turner; and each of which is incorporated herein by reference as if set forth in its respective entirety herein.

FIELD OF THE INVENTION

The present invention relates generally to the provision of information, and more particularly to the provision of informational, entertainment, educational, business and other audio and/or audio/visual presentations via a computer network.

BACKGROUND OF THE INVENTION

The Internet is a global network connecting millions of computers and linking users in more than 100 countries into exchanges of data, news and opinions. Unlike online services, which are centrally controlled, the Internet is decentralized. Each Internet enabled computer is independent, such that its user can choose which Internet services to use and which local services to make available to the global Internet community.

There are many types of content available via the Internet, including textual content, graphical content, audio content and video content. The amount of content available via the Internet is virtually unlimited. Accordingly, it can prove difficult for a user of an Internet enabled computer to identify and locate content of a particular type and relating to a particular subject.

A popular solution to finding desired content is to use a publicly available search engine. A search engine searches documents for specified keywords and returns a list of documents where the keywords were found. Typically, a search engine utilizes a webcrawler to provide documents. An indexer then typically reads the webcrawler provided documents and creates an index based on the words contained in each document. Each search engine typically uses its own methodology to create indices such that, ideally, only meaningful results are returned for each query. This is not always true though due to the complex nature and nuances of human language and efforts by document authors or providers to fool or trick the indexer into ranking its documents above those of

2

others. Examples of conventional search engines include those made available via www.yahoo.com, www.google.com and www.search.com, all by way of non-limiting example only.

Accordingly, there is a need for a system and method of using the Internet as a global network to unite people with common interests. Such a system and method may be used as productivity tools for business, and to educate and entertain consumers.

SUMMARY OF THE PREFERRED
EMBODIMENTS

A method for tracking digital media presentations delivered from a first computer system to a user's computer via a network including: providing a corresponding web page to the user's computer for each digital media presentation to be delivered using the first computer system; providing a identifier data to the user's computer using the first computer system; providing an applet to the user's computer for each digital media presentation to be delivered using the first computer system, wherein the applet is operative by the user's computer as a timer; receiving at least a portion of the identifier data from the user's computer responsively to the timer applet each time a predetermined temporal period elapses using the first computer system; and, storing data indicative of the received at least portion of the identifier data using the first computer system; wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from the first computer system directly to the users computer independent of the first computer system; and wherein the stored data is indicative of an amount of time the digital media presentation data is streamed from the second computer system to the user's computer.

BRIEF DESCRIPTION OF THE DRAWINGS

Understanding of the present invention will be facilitated by consideration of the following detailed description of the preferred embodiments of the present invention taken in conjunction with the accompanying drawings, in which like numerals refer to like parts:

FIG. 1 illustrates a block diagram of a system of networked computers;

FIG. 2 illustrates an electronic document according to an embodiment of the present invention;

FIG. 3 illustrates an electronic document according to an embodiment of the present invention;

FIG. 4 illustrates a flow diagram of a process according to an embodiment of the present invention;

FIG. 5 illustrates a flow diagram of a process according to an embodiment of the present invention;

FIG. 6 illustrates a flow diagram of a process according to an embodiment of the present invention;

FIG. 7 illustrates a block diagram of a system of networked computers in conjunction with telecommunications devices according to an embodiment of the present invention;

FIG. 8 illustrates a flow diagram of a process according to an embodiment of the present invention;

FIG. 9 illustrates an electronic document according to an embodiment of the present invention; and

FIG. 10 illustrates a flow diagram of a process according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED
EMBODIMENTS

It is to be understood that the figures and descriptions of embodiments of the present invention have been simplified to

illustrate elements that are relevant for a clear understanding of the present invention, while eliminating, for the purpose of clarity, many other elements found in typical website and audio/visual content delivery systems and methods. Those of ordinary skill in the art may recognize that other elements and/or steps are desirable and/or required in implementing the present invention. However, because such elements and steps are well known in the art, and because they do not facilitate a better understanding of the present invention, a discussion of such elements and steps is not provided herein.

For non-limiting purposes of explanation only, "computer," as referred to herein, refers to a general purpose computing device that includes a processor. "Processor," as used herein, refers generally to a device including a Central Processing Unit (CPU), such as a microprocessor. A CPU generally includes an arithmetic logic unit (ALU), which performs arithmetic and logical operations, and a control unit, which extracts instructions (e.g., code) from memory and decodes and executes them, calling on the ALU when necessary. "Memory," as used herein, refers to one or more devices capable of storing data, such as in the form of chips, or other medium like magnetic or optical discs. Memory may take the form of one or more random-access memory (RAM), read-only memory (ROM), programmable read-only memory (PROM), erasable programmable read-only memory (EPROM), or electrically erasable programmable read-only memory (EEPROM) chips, by way of further non-limiting example only. Memory may be internal or external to an integrated unit including the processor. Memory may be internal or external to the computer. Such memory may store a computer program, e.g., code or a sequence of instructions being operable by the processor. Such a computer may include one or more data inputs. Such a computer may include one or more data outputs. The code stored in memory may cause the processor, when executed by the processor, to set an output to a value responsively to a sensed input.

One type of computer executable code typically stored in memory so as to be executable by an Internet enabled computer is a browser application. For non-limiting purposes of explanation only, "browser application" or "browser," as used herein, generally refers to computer executable code used to locate and display web pages. Commercially available browsers are Microsoft Internet Explorer, Netscape Navigator, Apple Safari, Google Chrome and Firefox, which all support text, graphics and multimedia information, including sound and video (sometimes through browser plug-in applications). "Plug-in," as used herein, generally refers to computer executable code that adds a specific feature or service to a larger system, in the case of a browser plug-in, the browser application.

The terms "computer," "computer device and/or "computer system" as used herein may generally take the form of single computing devices or collections of computing devices having a common operator or under common control.

According to certain embodiments of the present invention, content may be aggregated for presentation to users. According to certain embodiments of the present invention, audio content may be aggregated for presentation to users. According to certain embodiments of the present invention, video content may be aggregated for presentation to users. According to certain embodiments of the present invention, audio and video content may be aggregated for presentation to users.

Referring now to FIG. 1, there is shown a block diagram of a system of networked computers 10. The illustrated system 10 includes a plurality of user computers 20, a plurality of

network server computers 30 and a network 40 interconnecting computers 20, 30 together.

Illustrated system 10 includes personal computing devices 22 and a personal digital assistant computer/web-enabled cell phone computer 24 by way of non-limiting example only. Communication links 26 communicatively couple devices 20 with network 40. Links 26 may take the form of wired and/or wireless communications links, including fiber optic, POTS, DSL, cable and/or multiple access or GSM based wireless telephony or data communications systems, for example. Network 40 may include portions of proprietary and service provider networks, as well as the Internet, for example. Illustrated system 10 includes a database server 32, a content or web server 34 and a file server 36, all by way of non-limiting example only. Communication links 26 communicatively couple devices 30 with network 40 as well. "Server", as used herein, generally refers to a computing device communicatively coupled to a network and that manages network resources. A server may refer to a discrete computing device, or may refer to an application that is managing resources rather than the entire computing device.

Referring now also to FIG. 2, there is illustrated a web page 200 according to an embodiment of the present invention. Web page 200 may be provided to computers 20 by computers 30 via network 40. Illustrated web page 200 aggregates audio and/or video content for presentation to users of computers 20.

Referring still to FIG. 2, the particularly illustrated web page 200 includes a category selector 205, a ranking selector 210, a new content indicator 215, a content type indicator 220, a page selector 225, particular content graphics 230, particular content type indicators 235 and particular content information 240 organized under a browser tab 245. Web page 200 may take other forms and/or present different content as is conventionally achieved in the pertinent arts.

Particular content graphics 230, particular content type indicators 235 and particular content information 240 are organized to indicate individual presentations. In the illustrated embodiment, presentations 265, 270, 275, are respectively shown. A user may select such a presentation for display by selecting an individual presentation for streaming or downloading, such as by clicking on an indicator 235, 240 or 245. For non-limiting purposes of explanation, "streaming," as used herein, generally refers to a technique for transferring data such that it can be processed as a substantially steady or continuous stream and a user's browser or plug-in can start presenting the data before the entire file has been transmitted. For non-limiting purposes of explanation, "downloading," as used herein, generally refers to a technique for transmitting data (e.g., an entire data file) between computers, such as between file server 36 (FIG. 1) and a computing device 22 (FIG. 1). In certain embodiments of the present invention, a commercially available content (e.g., audio and/or video podcast) delivery application, such as the Flash product available from Adobe Systems Inc., may be used to provide selected presentations to users' computers 20 (FIG. 1).

Referring still to FIGS. 1 and 2, a user of a device 20 may request page 200 from content server 34 using a browser application in a conventional manner. Server 34 may provide page 200 to the requesting computer 20 in a conventional manner, optionally using database server 32 to populate page 200, for example.

In certain embodiments of the present invention, when a user selects a category in selector 205, content server 34 may request database server 32 identify which presentations should be used to populate page 200 according to the selected category. Server 34 may then provide such a populated page

5

200 to the requesting user computer 20. Examples of categories that may be included and selected using selector 205 include art, autos and vehicles, bloggers and people, celebrity gossip, comedy, education, gadgets, health, how to and DIY, legal, music, news, and pets and animals, for example. By selecting one of these categories, a user may receive pages 200 populated with content according to the selected category.

In certain embodiments of the present invention, when a user selects a ranking in indicator 210, content server 34 may request database server 32 identify which presentations should be used to populate page 200 according to the selected ranking. Server 34 may then provide such a populated page 200 to the requesting user computer 20. Examples of rankings that may be included and selected using indicator 210 include most recent presentations and most popular presentations, for example. By selecting one of these rankings, a user may receive pages 200 populated with content according to the selected ranking.

In certain embodiments of the present invention, a user may select a populated presentation (e.g., 265, 270 or 275, FIG. 2). In response thereto, server 34 may request file server 36 either stream or download the selected presentation to the requesting user's computer 20, such as via a web page 200 in a conventional manner.

Referring now to FIG. 3, there is shown a view of web page 200 when tab 250 is selected. In the illustrated embodiment of FIG. 3, web page 200 includes a text box 255 and search button 260 under tab 250. In certain embodiments of the present invention, when tab 250 is selected, text box 255 and search button 260 may be presented on the user's computer 20 by server 34. A user may enter a search term into window 255 in a conventional manner. A user may then activate search button 260 in a conventional manner. Responsively thereto, content server 34 may request database server 32 identify which presentations should be used to populate page 200 according to the entered search term(s). Server 34 may then provide such a populated page 200 to the requesting user computer 20.

As will be appreciated by those possessing an ordinary skill in the pertinent arts, there are a number of ways to aggregate and provide content using web page 200.

In certain embodiments of the present invention, users may be permitted to directly upload and enter information regarding content, e.g., to file server 36 (FIG. 1). In certain embodiments of the present invention, users may be permitted to link presentations housed elsewhere in memory so as to be accessible to a computer 20 (FIG. 1) via network 40 (FIG. 1)—essentially registering them with database server 32 (FIG. 1). In certain embodiments of the present invention, presentations may be created using computers 20, 30. And, in certain embodiments of the present invention, presentations housed elsewhere in memory so as to be accessible to a computer 20 (FIG. 1) via network 40 (FIG. 1) may be automatically linked to—essentially registering them with database server 32 (FIG. 1).

Referring now also to FIG. 4, there is shown a flow diagram of a process 400 according to an embodiment of the present invention. Process 400 is suitable for permitting users to directly upload and enter information regarding content. Process 400 commences with a user providing log on information using a computer 20 at block 405, which is provided to computers 30 via network 40, in certain embodiments to server 34. Computers 30 log the user on at block 410, and communicates this status to the user via network 40, in certain embodiments by serving a page 200 (FIGS. 2, 3) to the logged on user's computer 20.

6

At block 415, the logged on user requests to upload content, e.g., by interacting in a conventional manner with web page 200. This request is provided to computers 30 via network 40. At block 420, computers 30 request information regarding the content to be uploaded. In certain embodiments, the requested information may include a content title, date, series information and description, akin to that to be displayed in a corresponding indicator 240 (FIGS. 2, 3). The request may further include a file identifier and location of the content indicative file to ultimately be uploaded. This request may be communicated to the user's computer 20 via network 40.

At block 425, the user provides at least a portion of the requested information, which is communicated to computers 30 via network 40. Some or all of the information provided may be screened or filtered or verified in conventional manners at block 430. In certain embodiments of the present invention, information provided at block 425 may be received and screened or filtered or verified at block 430 using web server 34. All or a portion of that information may then be stored using database server 32, for later use in populating web pages 200, for example.

At block 430, computers 30 indicate the received information is suitable for use and confirms the content may be uploaded. This indication is provided to the user's computer 20 via network 40. At block 435, the user's computer transmits the content to computers 30 via network 40, e.g., performs a file upload in a conventional manner. The content is received by computers 30 at block 440. In certain embodiments of the present invention, content transmitted and received at blocks 435, 440 may take the form of media file suitable for use as a podcast, for example. Such a file may be received by server 34 for example, and provided to server 36 for storage 450 and later retrieval for downloading and/or streaming pursuant to a user's interaction with webpage 200 (FIGS. 2, 3), for example. In such a case, server 32 may associate the stored content indicative information provided at block 425 with the file stored at block 450.

Referring now also to FIG. 5, there is shown a flow diagram of a process 500 according to an embodiment of the present invention, Process 500 is suitable for permitting users to link presentations housed elsewhere in memory so as to be accessible to a computer 20 via network 40.

Process 500 commences with a user providing log on information using a computer 20 at block 505, which is provided to computers 30 via network 40, in certain embodiments to server 34. Computers 30 log the user on at block 510, and communicate this status to the user via network 40, in certain embodiments by serving a page 200 (FIGS. 2, 3) to the logged on user's computer 20.

At block 515, the logged on user requests to link or register content, e.g., by interacting in a conventional manner with web page 200. This request is provided to computers 30 via network 40. At block 520, computers 30 request information regarding the content to be linked. In certain embodiments, the requested information may include a content title, date, series information and description, akin to that displayed in a corresponding indicator 240 (FIGS. 2, 3). The request may further include a file identifier and location of the content indicative file to be linked. This request may be communicated to the user's computer 20 via network 40.

At block 525, the user provides at least a portion of the requested information, which is communicated to computers 30 via network 40. Some or all of the information provided may be screened or filtered or verified in conventional manners at block 530. In certain embodiments of the present invention, information provided at block 525 may be received

and screened or filtered or verified at block 530 using web server 34. In certain embodiments of the present invention, the file location data (e.g., an Internet address at which the file is available) may be checked to see if a valid media file is located thereat. All or a portion of that information may then be stored using database server 32, for later use in populating web pages 200, for example.

At block 530, computers 30 indicate the received information is suitable for use and confirms the content may be linked. At block 550 the received information may be stored using server 32 for later retrieval and use. Server 32 may also associate the linked content indicative information provided at block 525 with the file address stored at block 550.

Certain embodiments of the present invention may provide the ability to track the number of visitors to the platform of the present invention, and additionally the number of visitors per content via the platform of the present invention. Further, the number of pages viewed by each visitor may additionally be tracked, such as in a tabular format, and such information may be continuously updated for as long as a user remains on a given page, that is, for as long as a user continues to watch a particular show. For example, it may be determined when a user begins and ends listening to and/or watching a presentation, e.g., a podcast, for example. Where a selected presentation is streamed from computers 30, such an inquiry may be relatively simple, by confirming the content streaming is progressing as expected, for example. Where content is housed elsewhere and linked to by computers 30, such a direct inquiry may not be readily available though. Tracking may be performed, for example, via entry into one or more tables of database server 32 of timed data. At each expiration of a timer, such as every 15 seconds, a table entry may be made corresponding to the user, the page the user is on, and, to the extent the user is on the same page as was the user upon the last expiration of the timer, the user's total time, to the current time, spent on that same page. The user may be identified by, for example, any of a number of known methodologies, such as the information the user used to login, the user's IP address, the user's response to an identifying query, or the like.

Thus, certain embodiments of the present invention provide a capability to know that a viewer began viewing a particular show at a certain time, and when a user began viewing a different page, or show, thereby providing knowledge of how long a particular viewer spent on a particular page. Such knowledge is not conventionally available, and the provision of such knowledge by certain embodiments of the present invention allows for an increasing scale of payments for advertising displayed on a given page correspondent to how long a viewer or viewers remain, or typically remain, on that particular page or like pages. Thus, a tabular tracking of the present invention allows for the knowledge of how long a viewer spends on a page, what the viewer was viewing or listening to on the given page, the ads shown while the viewer was viewing or listening, how long the ads were shown, and what ads were shown to the view correspondent to that viewer's identification and/or login.

Referring now also to FIG. 6, there is shown a flow diagram of a process 600 according to an embodiment of the present invention. Process 600 is suitable for permitting users to create presentations, such as by hosting an audio show that may be recorded to create a podcast, using computers 20, 30.

Process 600 commences with a user providing log on information using a computer 20 at block 605, which is provided to computers 30 via network 40, in certain embodiments to server 34. Computers 30 log the user on at block 610, and

communicate this status to the user via network 40, in certain embodiments by serving a page 200 (FIGS. 2, 3) to the logged on user's computer 20.

At block 615, the logged on user requests to create content or host a show, e.g., by interacting in a conventional manner with web page 200. This request is provided to computers 30 via network 40. At block 620, computers 30 request information regarding the content to be created. In certain embodiments, the requested information may include a content title, date, series information and description, akin to that displayed in a corresponding indicator 240 (FIGS. 2, 3). The request may further include a phone number at which the user may be reached. This request may be communicated to the user's computer 20 via network 40.

At block 625, the user provides at least a portion of the requested information, which is communicated to computers 30 via network 40. Some or all of the information provided may be screened or filtered or verified in conventional manners at block 630. In certain embodiments of the present invention, information provided at block 625 may be received and screened or filtered or verified at block 630 using web server 34. In certain embodiments of the present invention, the user's phone number may be checked to see if it is valid. All or a portion of that information may then be stored at block 635 using database server 32, for later use in populating web pages 200, for example.

At block 640, the requesting user indicates he would like to begin creating the presentation, e.g., by interacting in a conventional manner with web page 200. This indication is communicated to computers 30 via network 40. At block 645 computers 30 initiate a new presentation creation session. At block 650, a voice communications session between computers 30 and the user is commenced. In certain embodiments of the present invention, a telephone call may be automatically placed by computers 30 at block 650 to the phone number indicated at block 625.

Referring now to FIG. 7, there is shown a block diagram of a system of networked computers and telephones 700. Like system 10, illustrated system 700 includes personal computing devices 22 and a personal digital assistant/web-enabled cellular phone computer 24 by way of non-limiting example only. Communication links 26 communicatively couple devices 20 with network 40. Links 26 may take the form of wired and/or wireless communications links, including fiber optic, POTS, OSL, cable and/or multiple access or GSM based wireless telephony or data communications systems, for example. Network 40 may include portions of proprietary and service provider networks, as well as the Internet, for example. Illustrated system 10 includes a database server 32, a content or web server 34 and a file server 36, all by way of non-limiting example only. Communication links 26 communicatively couple devices 30 with network 40 as well.

System 700 additionally includes conventional telephone 705 associated with (as indicated by label 720) a particular computing device 22, e.g., by both corresponding to a given requesting user, for example. In the illustrated embodiment, phone 705 may be communicatively coupled to computers 30 independent of network 40 (e.g., via 725). In the illustrated embodiment, phone 705 may be communicatively coupled to computers 30 via network 40 (e.g., link 710). In certain embodiments of the present invention phone 705 may take the form of a POTS phones. In certain embodiments of the present invention phone 705 may take the form of a VoIP phone. In certain embodiments of the present invention, phone 705 may take the form of a cellular phone. In certain embodiments of the present invention, phone 705 is independent of the associated computer 22. In certain embodiments

of the present invention, phone **705** may be communicatively coupled to computers **30** independent of any connection between the associated computer **22** and computers **30**.

Referring still to FIGS. **6** and **7**, a requesting user may be called at block **650** by computers **30** placing a conventional telephone call to the phone number provided at block **625**. Upon the call being answered using phone **705**, a pre-recorded audio message indicating the content will be created may be played. Thereafter, the requesting user, or his designee for example, may speak into phone **705**, thereby hosting a show, for example. Responsively thereto, computers **30** may digitize the spoken show and store a media file indicative of it (e.g., using file server **36**), as indicated at block **655**.

Information provided at block **625** and stored at block **635** may include identifications of intended audience members for the presentation, e.g., an audience for the show to be hosted. This additional information may be used at block **660** to initiate analogous telephone calls to those numbers as well. In this way, a phone audience may hear the show live at a plurality of locations. For non-limiting purposes of explanation, this is shown in FIG. **7** as phone **730**, which is associated with computer **24** as designated by label **740**.

Such a “dial out” functionality allows for an understanding of where the user/viewer/listener can be reached, located, and/or may allow for a myriad additional features in the present invention. For example, a pinpoint geographic location of broadcast listeners may be placed on a map, such as via website **200** to thereby illustrate where other listeners of the broadcast are specifically located. Such a mapping functionality may be realized using a commercially available mapping application, such as Google Maps, for example.

In certain embodiments of the present invention, shows may be streamed analogously as described above as they are being recorded, for example.

It should further be understood such a content generation functionality provides additional advantages. For example, enhanced telephone conferences may be readily achieved according to certain embodiments of the present invention. Such enhanced conferences may exhibit an automatic dial out to conference attendees, including the host and audience. Such enhanced conferences may exhibit automatic recording and archival for later playback as a podcast, for example. Such enhanced functionalities may advantageously be achieved without the host having access to any particular resources other than a general purpose Internet enabled computer and a conventional telephone. Such enhanced functionalities may advantageously be achieved without the any audience member having access to any particular resources other than a conventional telephone. Accordingly, enhanced telephone conferencing may be readily achieved.

In certain embodiments of the present invention, certain portions of aggregated content may have access thereto restricted to authorized members. For example, information provided at blocks **425**, **525** and/or **625** may include an authorized group identifier or content password. Such an identifier and/or password may be stored using database server **32**. When a user seeks to playback such protected content, e.g., by interacting with web page **200** as set forth above, the user may need to log in (e.g., analogously to log in at blocks **405**, **410**, **505**, **510**, **605**, **610**) or provide the corresponding password. Where a group identifier is used, database server **32** may indicate what groups a logged in user is authorized for, so as to selectively permit access to protected content to authorized users. Such groups may, by way of non-limiting example only, include businesses and other private organizations.

Referring now also to FIG. **8**, there is shown a flow diagram of a process **800** according to an embodiment of the present

invention. Process **800** is suitable for automatically aggregating and linking to presentations housed elsewhere in memory so as to be accessible to a computer **20** (FIG. **1**) via network **40** (FIG. **1**)—essentially registering them with database server **32** (FIG. **1**).

Syndication of Internet content is becoming more commonplace. Really Simple Syndication (“RSS”) is a family of Internet feed formats used to publish content that may be frequently updated, such as podcasts (RSS 2.0). RSS utilizes a standardized format. An RSS document (sometimes referred to as a “feed,” “web feed” or “channel”) typically contains either a summary of content from an associated web site or the full text.

An RSS may itself be used to aggregate content from multiple web sources in one place. RSS content is typically accessed using an RSS reader application. Such an application may be a thin, web-page based application or a downloaded application executed on a user’s computer (e.g., **20**, FIG. **1**). RSS feeds may typically be subscribed to by entering or selecting the feed’s link using the reader. The RSS reader typically checks the user’s subscribed feeds for new content at predetermined intervals, downloads updates, and provides a user interface to monitor and view the feeds.

Embodiments of the present invention will be discussed with regard to RSS 2.0 feeds for non-limiting purposes of explanation only. It should be recognized that embodiments of the present invention may be suitable for use with other types of content (e.g., audio/video) feeds.

Referring again to FIG. **8**, process **800** commences with a user providing log on information using a computer **20** at block **805**, which is provided to computers **30** via network **40**, in certain embodiments to server **34**. Computers **30** log the user on at block **810**, and communicate this status to the user via network **40**, in certain embodiments by serving a page **200** (FIGS. **2**, **3**) to the logged on user’s computer **20**.

At block **815**, the logged on user requests to link an RSS feed, e.g., by interacting in a conventional manner with web page **200**. This request is provided to computers **30** via network **40**. At block **820**, computers **30** request information regarding the content to be created. In certain embodiments, the requested information may include a content title, series information and description, akin to that displayed in a corresponding indicator **240** (FIGS. **2**, **3**). The request may further include RSS feed identification and/or access information through which the feed may be accessed. This request may be communicated to the user’s computer **20** via network **40**.

At block **825**, the user provides at least a portion of the requested information, which is communicated to computers **30** via network **40**. Some or all of the information provided may be screened or filtered or verified in conventional manners at block **830**. In certain embodiments of the present invention, information provided at block **825** may be received and screened or filtered or verified at block **830** using web server **34**. In certain embodiments of the present invention, the feed identifier and/or access information may be checked to see if it is valid. All or a portion of that information may then be stored at block **850** using database server **32**, for later use in populating web pages **200**, for example. At block **850**, the feed may further be accessed to acquire information regarding and/or either links to or the feed content itself then present. All of this information may be automatically aggregated using computers **30** in accordance with the methods described herein-above with regard to FIGS. **4** and/or **5**, where the feed information (e.g., RSS associated XML data) is used in lieu of user provided information. The date and time

when content is automatically acquired via such a registered RSS feed may also be stored at block **850** using computers **30**, e.g., database server **32**.

At block **855**, computers **30** may determine if new content exists for one or more feeds stored at block **850**. This may be accomplished in any of a number of conventional manner, including periodically checking when the feed was last updated and/or the content available there-through to data stored at block **850**. When new or changed content is found, the data stored at block **855** may be appended or amended to reflect the new content.

It should further be understood such a content acquisition provides additional advantages. For example, each user wishing to identify and view content available via an RSS feed may conventionally need to obtain and operate an RSS reader application. Further, each such RSS reader application would need to access each identified RSS feed. This leads to substantial bandwidth usage, for example. In contrast, certain embodiments of the present invention permit a user to access RSS content without the need for his own RSS reader. Further, embodiments of the present invention only require that system **30** access each RSS feed, as opposed to each system **30** user computer **20** wishing to access the RSS feeds, leading to substantial savings in network resources. Further, certain embodiments of the present invention allow user to access and compare content available via RSS feeds they are not even aware of, e.g., by their interaction with webpage **200** as discussed above, where webpage **200** includes content added using the methodology of process **800**, for example. Accordingly, certain embodiments of the present invention provide for enhanced content syndication and aggregation, as compared to even RSS feeds themselves, for example. And, certain embodiments of the present invention provide for automatic aggregation of RSS fed content in combination with non-RSS fed content in a single application independent of any user RSS reader application.

In certain embodiments of the present invention, web page views and/or web site visits (e.g., sessions) may be tracked. A page view, as used herein, generally refers to a request made to a web server for a web page, as opposed to just a page component, such as a graphic, for example. A visit, as used herein, generally refers to a sequence of web page and/or component requests from a particular user's computer, within some predetermined period of time. Commercially available server log file analysis applications may be used to gather such information, for example.

In certain embodiments of the present invention, more detailed tracking information may be desired. For example, it may be desirable to know not only that a certain number of users requested and accessed certain presentations, but also how long a user actually watched, and/or listened, to a presented program, after selection via webpage **200** (FIGS. **2**, **3**), for example. Certain embodiments of the present invention may provide the ability to track the number of visitors to the platform of the present invention, and additionally the number of visitors per content via the platform of the present invention, and additionally information regarding how long presentations were watched and/or listened.

For example, and referring now to FIG. **9**, there is shown a view of a web page **900** according to an embodiment of the present invention. Web page **900** generally includes portions **910**, **920**, **930** and **940**. Web page **900** may be provided to a user's computer **30** responsively to user selection of a presentation shown on a populated web page **200** (FIG. **2**). By way of non-limiting explanation, should a user viewing web page **200** (FIG. **2**) select a presentation **265** for viewing and/or listening, a suitably populated web page **900** may be served

by computers **20**. In such a served web page **900**, portion **930** may be utilized to playback the selected presentation in a conventional manner, e.g., by downloading the content into or streaming the content to a media player application or plug-in. Portions **910**, **940** may be used to display related information, such as advertisements for example. In such a case, it may be desirable to be able to reliably identify how long the media was actually, or may typically be played, in order to appropriately value portions **910**, **920** as available advertising billboard space. By way of further, non-limiting, example, while a per-click or per-display pricing schedule for portions **910**, **940** may be used, where portion **920** is used to play-back content a typical user watches and/or listens to for ten minutes, portions **910**, **940** may be worth more than where content play-back is typically for less than thirty-seconds.

Where content is directly stored using an operator's system (e.g., computers or computer system **20**, FIG. **2**), such as by using the methodology of process **400** (FIG. **4**) or process **600** (FIG. **6**), such a tracking may be achieved by tracking requests from and pages viewed by each visitor, such as in a tabular format. As a system operator maintains control over the operation of system **30** in such a case, system **30** may be monitored to determine how long data is streamed therefrom, for example. Data indicative of this period, such as a presentation identifier and a value indicative of the time the presentation was actually streamed for, may be logged by system **30** (e.g., using database server **32**, for example). For example, it may be determined when a user begins and ends listening to and/or watching a presentation, e.g., a podcast, by tracking when a web page was loaded and for example by determining when streaming of data to such a loaded web page ceases. Where a selected presentation is streamed from computers **20**, such a methodology may be directly implemented by system **20**, by confirming the content streaming is progressing as expected, for example.

Where content is not uploaded to an operator's system (e.g., computers or computer system **20**, FIG. **2**) and is instead remotely stored from yet aggregated by system **30**, e.g., using the methodology of process **500** (FIG. **5**) or process **800** (FIG. **8**), for example, tracking may not be so straightforward. As an operator of system **30** does not necessarily exercise control over the content data storage resource, the operator may not be able to directly operate the storage resource in a manner to directly track how long content is streamed therefrom to a particular user.

In certain embodiments of the present invention, aggregated content playback may advantageously be tracked in a substantially same manner, regardless of whether it is streamed from system **30** or otherwise unrelated computer systems operated by third parties. In certain embodiments of the present invention, tracking information may be continuously or substantially continuously updated for as long as a user continues to watch or listen to a particular show, regardless of whether the content data is streamed from an operator's computer system **30** or a third party's computer system.

Referring now to FIG. **10**, there is shown a block diagram of a process **1000** according to an embodiment of the present invention. Process **1000** commences with a user's computer **20** receiving a web page from system **20** (FIG. **2**) at block **1010**. Such a received web page may take the form of page **900** (FIG. **9**), for example. As is shown in FIG. **9**, page **900** includes portion **930**, which may be used to play-back user selected content via his computer **20** and a suitable plug-in or media player, for example. As explained herein, data indicative of the content played using portion **920** may be supplied by system **30** or a third party's computer system. Regardless, page **900** may include a timer applet. "Applet," as used herein,

13

generally refers to a software component that runs in the context of another program, in the case of page 900 of FIG. 9, a web browser. Such an applet may typically be used to perform a specific function or task, usually narrow in scope. In the case of FIGS. 9 and 10, such a timer applet may be used to indicate when a pre-determined temporal period has elapsed. For example, such an applet may be used to indicate each time some temporal period, such as 10, 15 or 30 seconds, elapses. Such a timer applet may be started at block 1020.

At block 1030, when the applet determines the predetermined temporal period has elapsed, it signals its continued execution to system 20. In response, system 30 may log receipt of this indication, such as by using database server 32. In certain embodiments of the present invention, web page 900 (FIG. 9) may be accompanied with identifying data, such as in form of a cookie. A "cookie," as used herein, generally refers to a message provided to a web browser by a web server. The browser stores the message in a data or text file. In certain embodiments of the present invention, the applet may cause the cookie, or associated data, to be transmitted from the user's computer 20 to system 30, where upon receipt it, or data associated with it, may be logged, such as by using database server 32.

By way of further non-limiting example, at each expiration of temporal period as determined by the timer applet, such as every 15 seconds, a table entry may be made of the user, the page the user is on, and, to the extent the user is on the same page as was the user upon the last expiration of the timer, the user's total time, to the current time, spent on that same page using database server 32. The user may be identified by, for example, any of a number of known methodologies, such as the information the user used to login, the user's IP address, the user's response to an identifying query, or the like.

In certain embodiments of the present invention, the timer applet may cause data indicative of the total time spent on the web page presenting the presentation that has elapsed. In certain embodiments of the present invention, the timer applet may cause data indicative of another temporal cycle having passed while the web page presents the presentation. In the latter, a value indicative of the number of cycles that have passed in database 32 may be incremented each time the data is received, for example.

Thus, certain embodiments of the present invention provide the capability to know that a viewer began viewing a particular show at a certain time, and to know when a user began viewing a different page, or show, thereby providing knowledge of how long a particular viewer spent on a particular page. Such knowledge is not conventionally available, and the provision of such knowledge by certain embodiments of the present invention allows for an increasing scale of payments for advertising displayed on a given page corre-

14

spondent to how long a viewer or viewers remain, or typically remain, on that particular page or like pages. Thus, the tabular tracking of the present invention allows for the knowledge of how long viewer spends on a page, what the viewer was viewing or listening to on the given page, the ads shown while the viewer was viewing or listening, how long the ads were shown, and what ads were shown to the view correspondent to that viewer's identification and/or login.

Those of ordinary skill in the art may recognize that many modifications and variations of the present invention may be implemented without departing from the spirit or scope of the invention. Thus, it is intended that the present invention covers the modifications and variations of this invention provided they come within the scope of the appended claims and their equivalents.

What is claimed is:

1. A method for tracking digital media presentations delivered from a first computer system to a user's computer via a network comprising:

providing a corresponding web page to the user's computer for each digital media presentation to be delivered using the first computer system;

providing identifier data to the user's computer using the first computer system;

providing an applet to the user's computer for each digital media presentation to be delivered using the first computer system, wherein the applet is operative by the user's computer as a timer;

receiving at least a portion of the identifier data from the user's computer responsively to the timer applet each time a predetermined temporal period elapses using the first computer system; and

storing data indicative of the received at least portion of the identifier data using the first computer system;

wherein each provided webpage causes corresponding digital media presentation data to be streamed from a second computer system distinct from the first computer system directly to the user's computer independent of the first computer system;

wherein the stored data is indicative of an amount of time the digital media presentation data is streamed from the second computer system to the user's computer; and

wherein each stored data is together indicative of a cumulative time the corresponding web page was displayed by the user's computer.

2. The method of claim 1, wherein the storing comprises incrementing a stored value dependently upon the receiving.

3. The method of claim 2, wherein the received data is indicative of a temporal cycle passing.

* * * * *

APPENDIX B-2



US005796952A

United States Patent [19]

[11] Patent Number: 5,796,952

Davis et al.

[45] Date of Patent: Aug. 18, 1998

[54] METHOD AND APPARATUS FOR TRACKING CLIENT INTERACTION WITH A NETWORK RESOURCE AND CREATING CLIENT PROFILES AND RESOURCE DATABASE

[75] Inventors: Owen Davis, New York; Vidyut Jain, Brooklyn, both of N.Y.

[73] Assignee: Dot Com Development, Inc., New York, N.Y.

[21] Appl. No.: 821,534

[22] Filed: Mar. 21, 1997

[51] Int. Cl.⁶ G06F 13/00

[52] U.S. Cl. 395/200.54

[58] Field of Search 364/DIG. 1 MS File, 364/DIG. 2 MS File; 380/4; 395/200.3, 200.31, 200.32, 200.33, 200.54, 280, 381, 670, 680, 712

[56] References Cited

U.S. PATENT DOCUMENTS

| | | | |
|-----------|---------|------------------|------------|
| 4,977,594 | 12/1990 | Shear | 380/4 |
| 5,638,443 | 6/1997 | Stefik et al. | 380/4 |
| 5,675,510 | 10/1997 | Coffey et al. | 364/514 A |
| 5,682,525 | 10/1997 | Bouve et al. | 395/615 |
| 5,706,502 | 1/1998 | Foley et al. | 395/682 |
| 5,708,780 | 1/1998 | Levergood et al. | 395/200.12 |
| 5,710,918 | 1/1998 | Lagarde et al. | 395/680 |
| 5,715,453 | 2/1998 | Stewart | 395/615 |

OTHER PUBLICATIONS

S. Gundavaram. *CGI Programming on the World Wide Web* (O'Reilly & Assoc., Inc.), pp. 202-204.

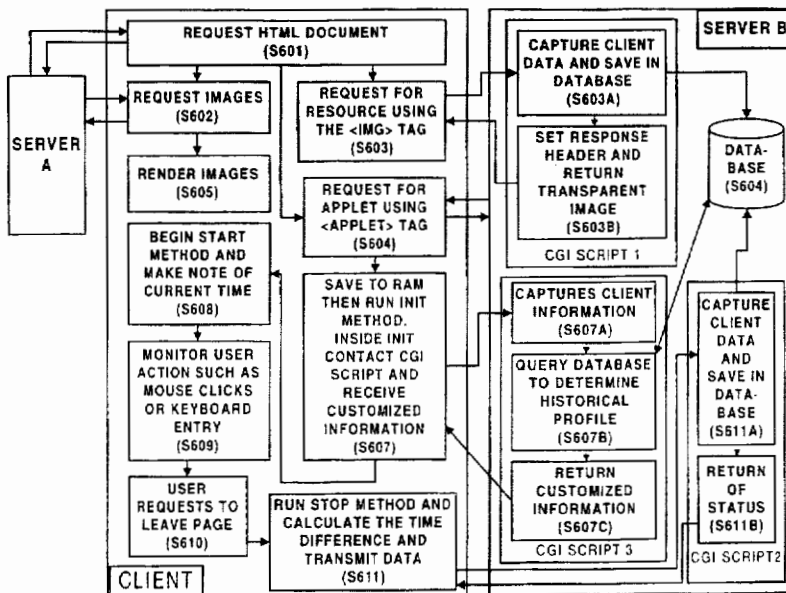
G. Cornell and S. Horstmann. *Core Java* (The Sunsoft Press), pp. 562-579.

Primary Examiner—Robert B. Harrell
Attorney, Agent, or Firm—Adams & Wilks

[57] ABSTRACT

A method for monitoring client interaction with a resource downloaded from a server in a computer network includes the steps of using a client to specify an address of a resource located on a first server, downloading a file corresponding to the resource from the first server in response to specification of the address, using the client to specify an address of a first executable program located on a second server, the address of the first executable program being embedded in the file downloaded from the first server, the first executable program including a software timer for monitoring the amount of time the client spends interacting with and displaying the file downloaded from the first server, downloading the first executable program from the second server to run on the client so as to determine the amount of time the client interacts with the file downloaded from the first server, using a server to acquire client identifying indicia from the client, and uploading the amount of time determined by the first executable program to a third server. The first executable program may also monitor time, keyboard events, mouse events, and the like, in order to track choices and selections made by a user in the file, and may execute upon the occurrence of a predetermined event, as well as monitoring or determining the amount of information downloaded by the client. The monitored information and client identifying indicia is stored on a database in a server for use in analysis and for automatically serving out files assembled according to user interests and preferences.

71 Claims, 7 Drawing Sheets



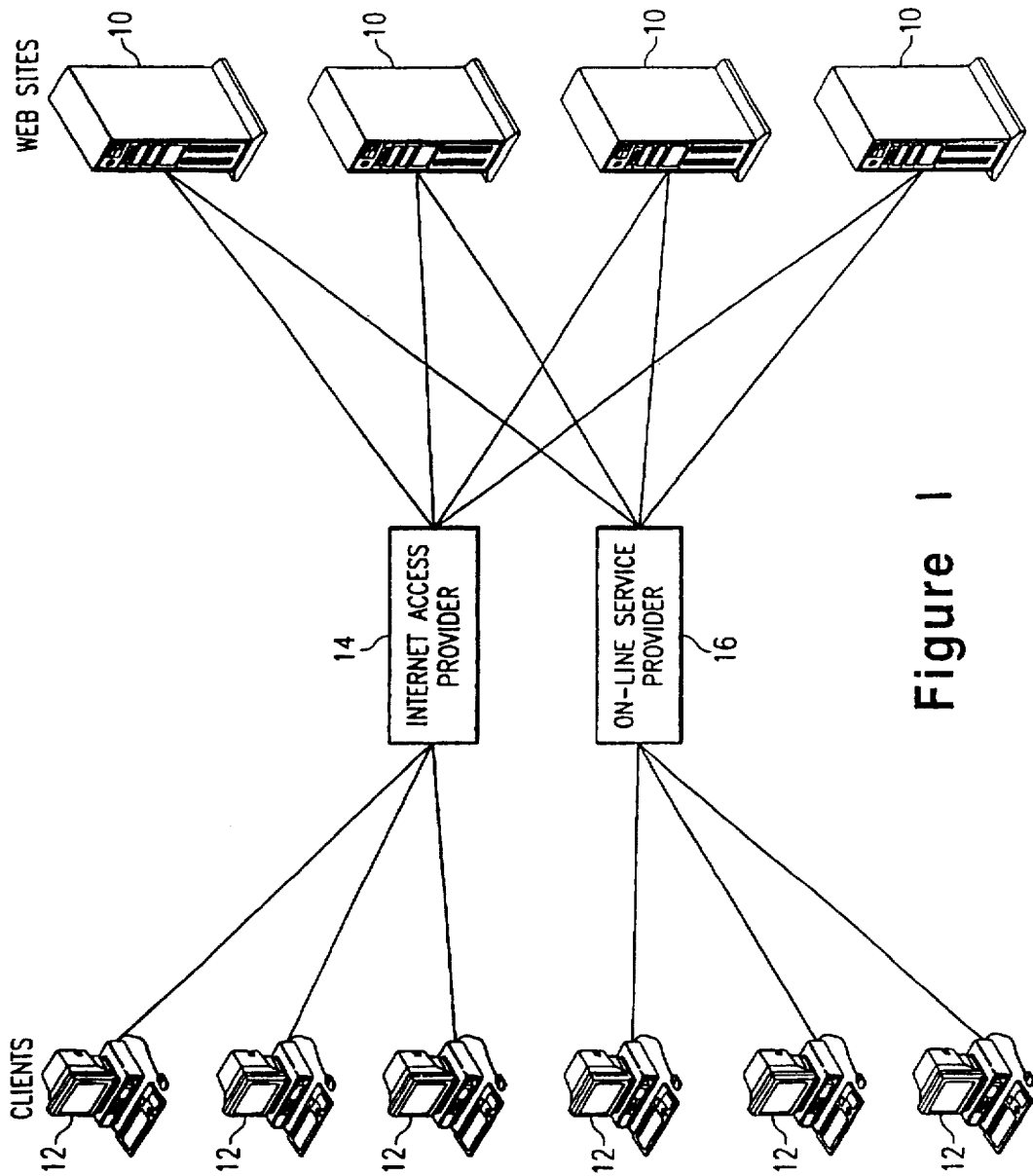
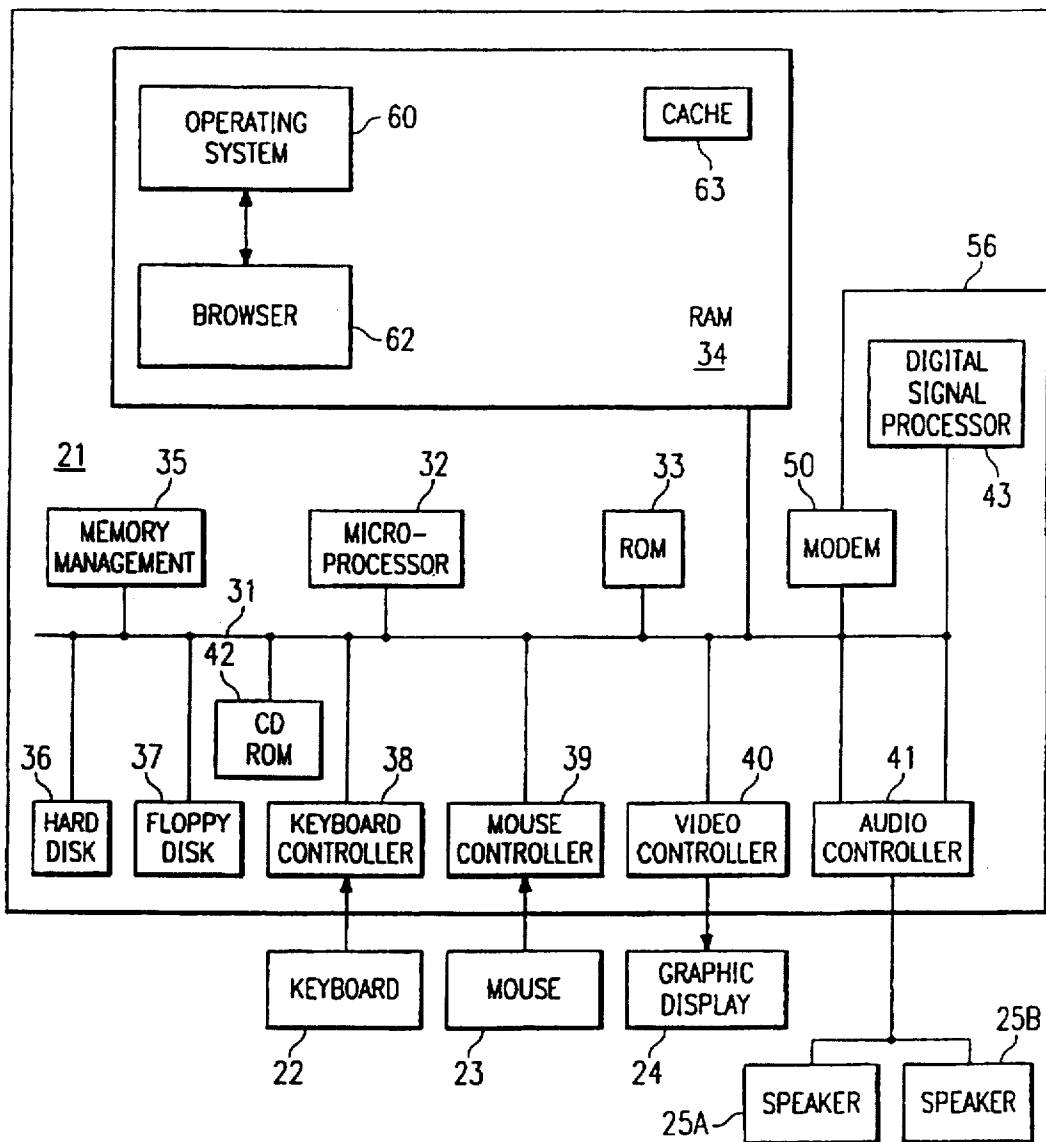


Figure 1

Figure 2



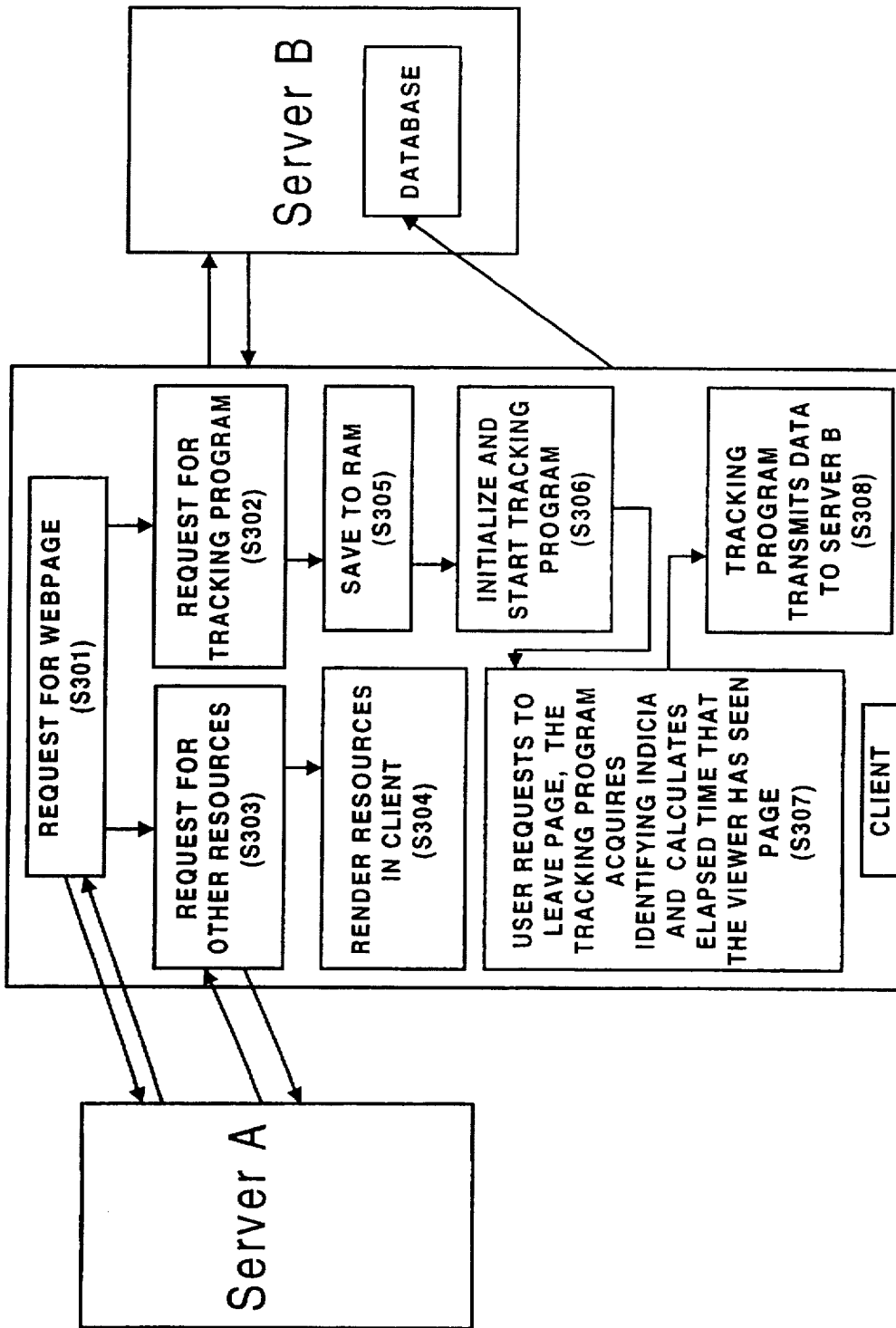


Figure 3

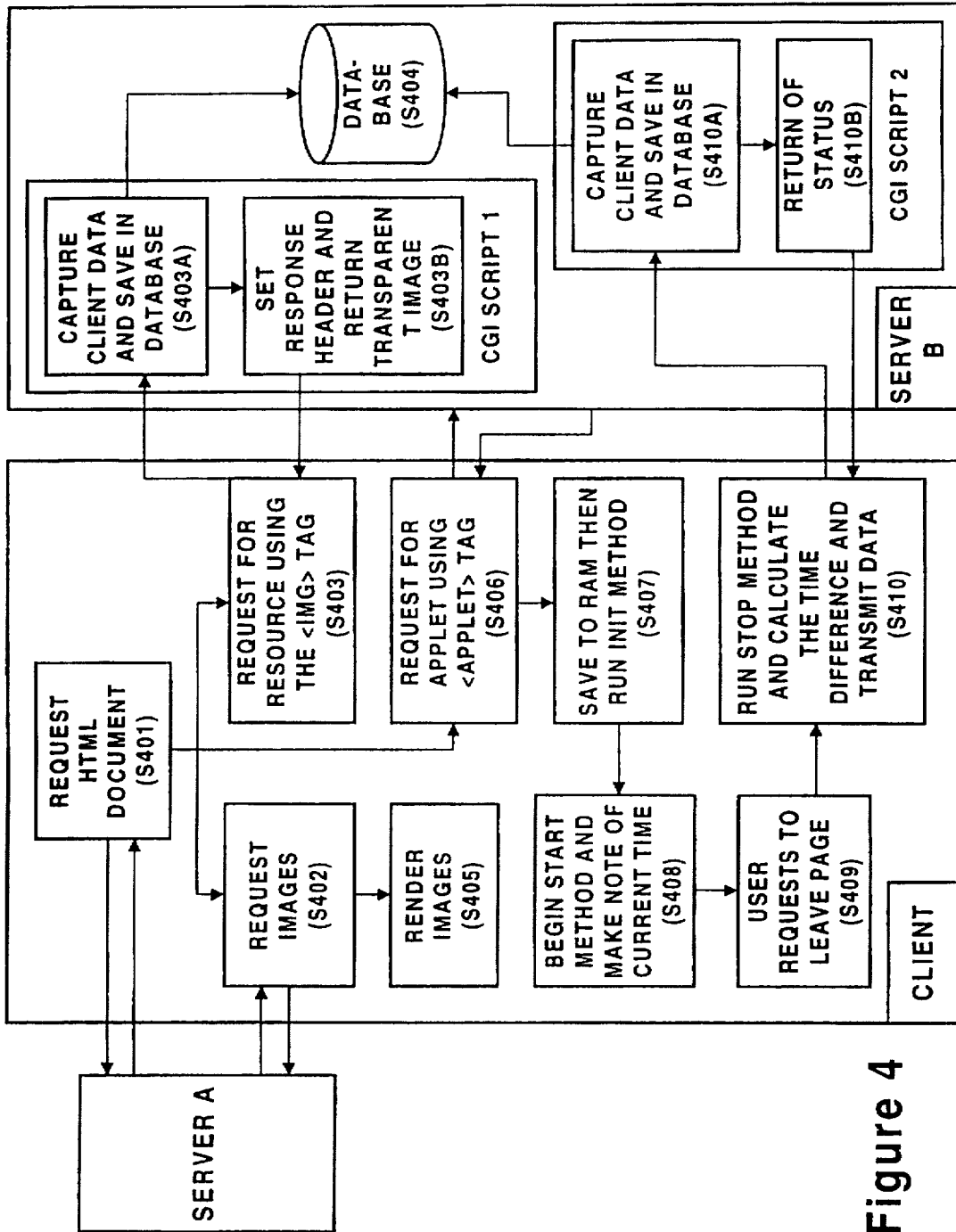


Figure 4

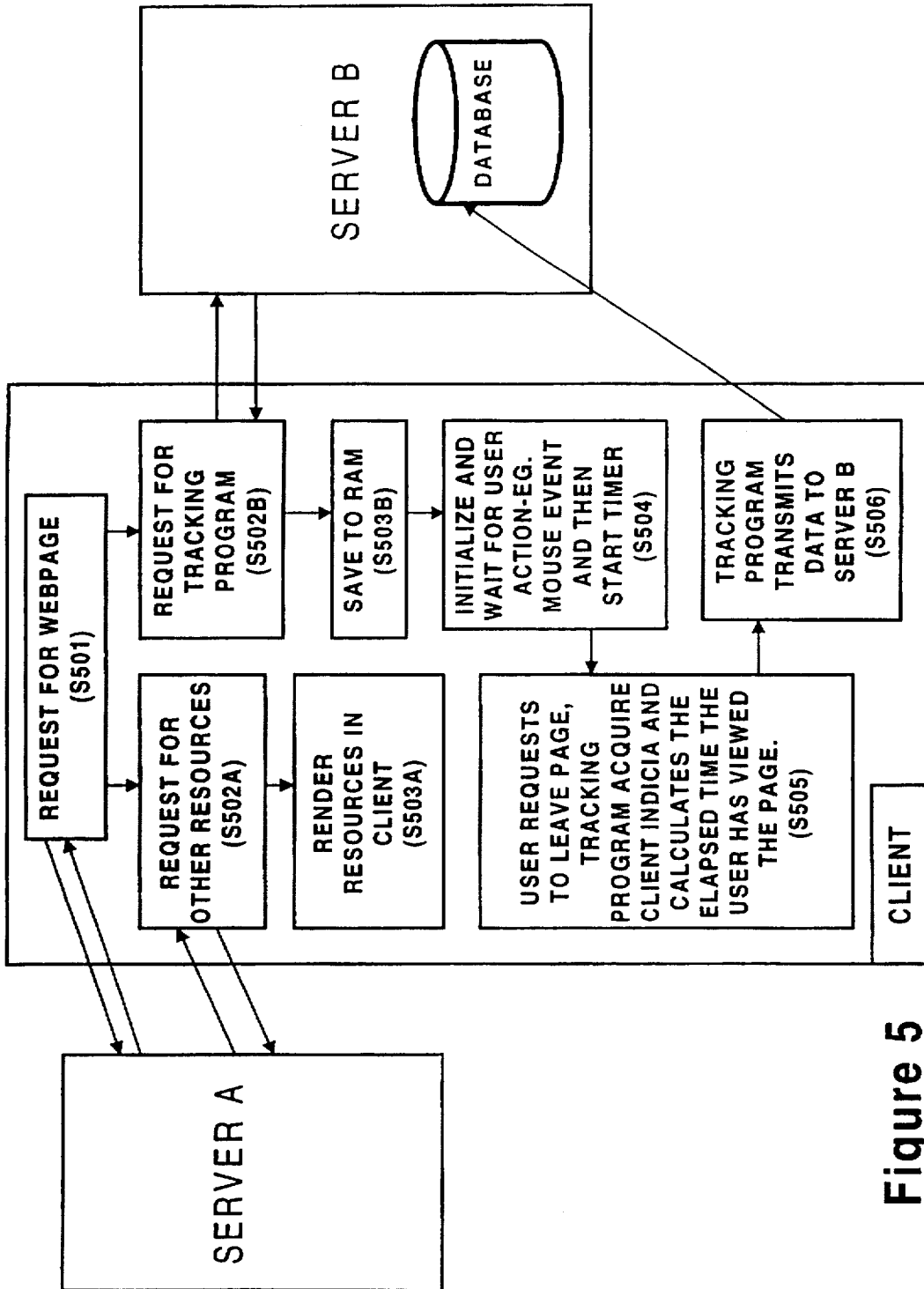


Figure 5

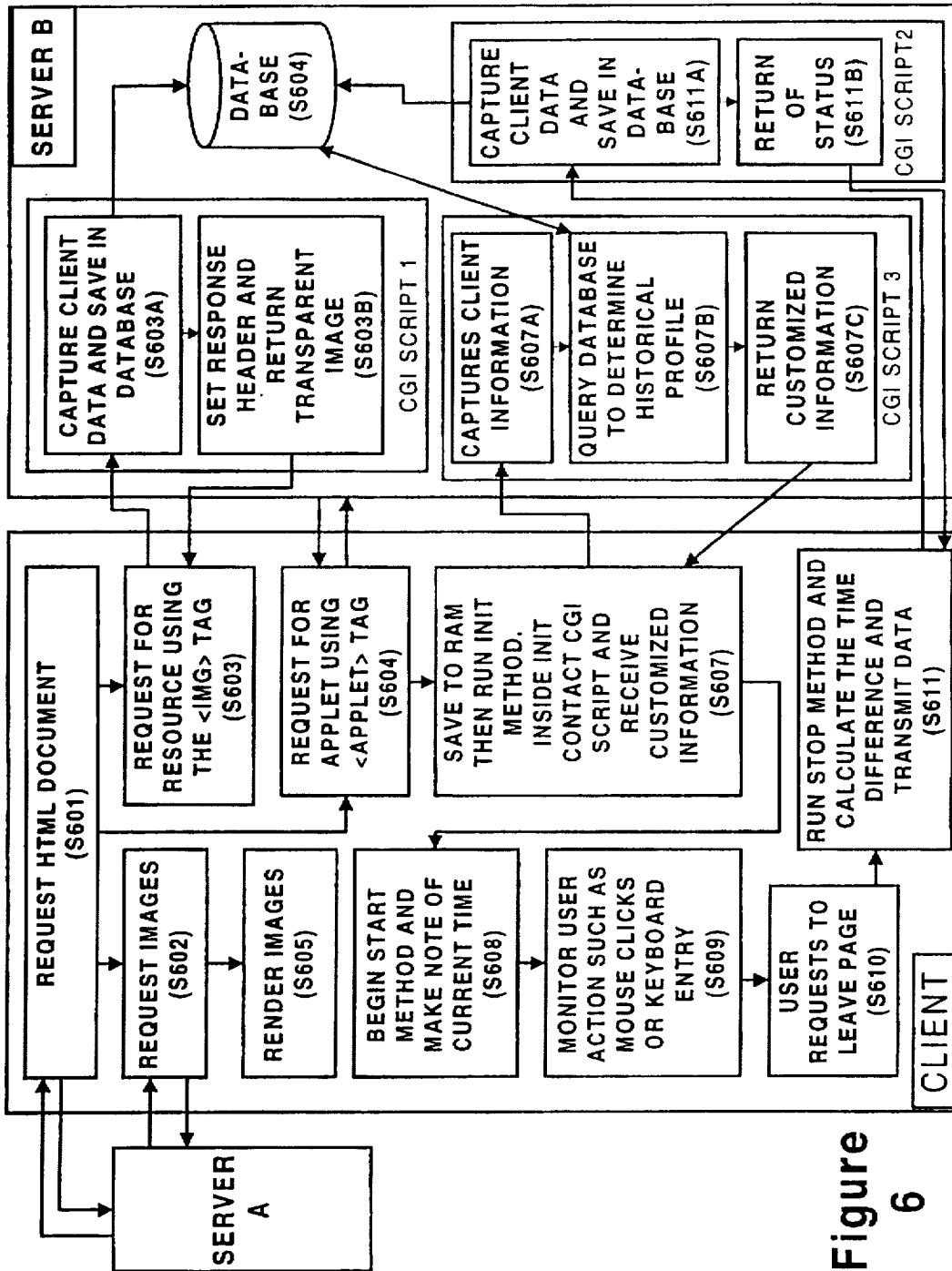


Figure 6

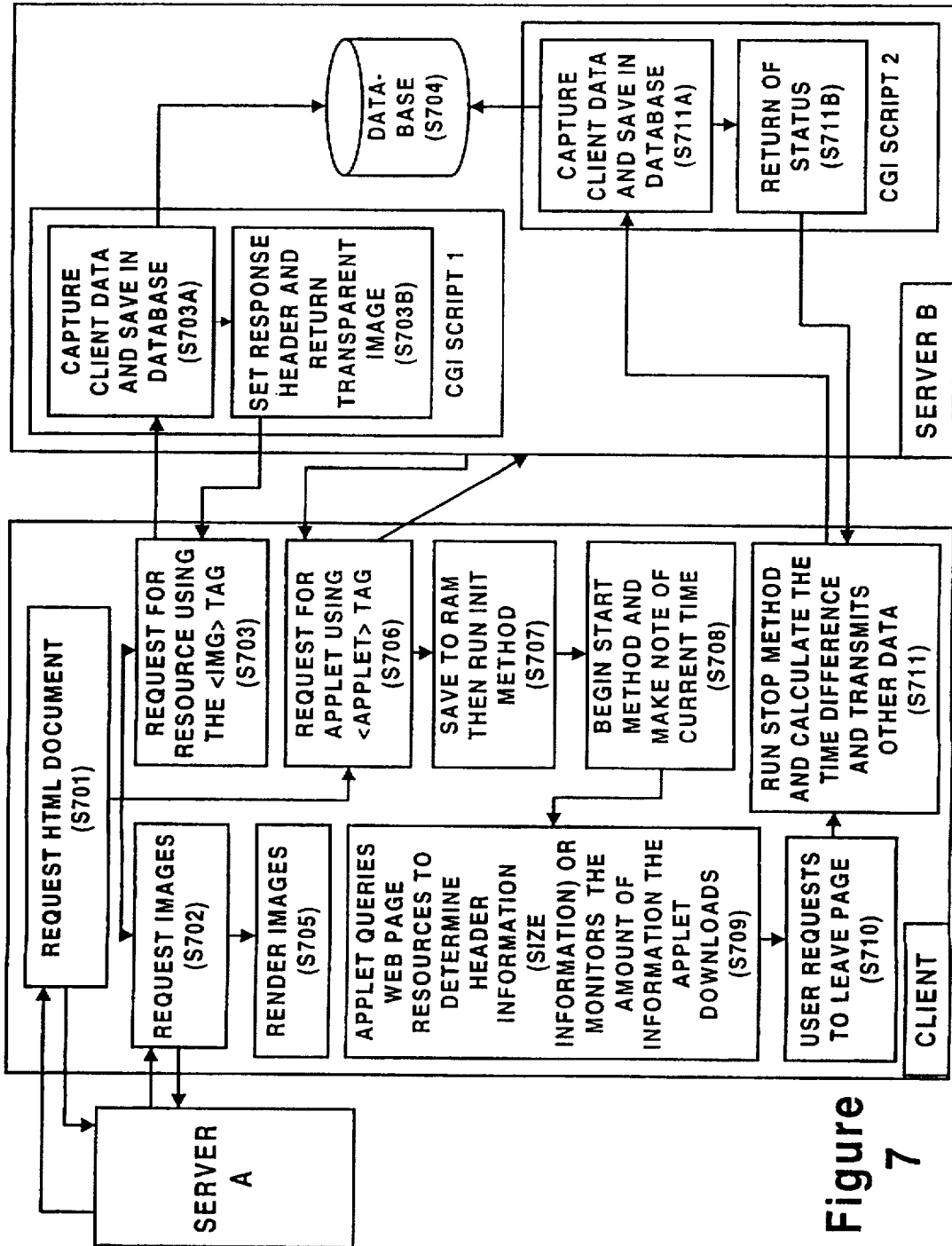


Figure 7

**METHOD AND APPARATUS FOR
TRACKING CLIENT INTERACTION WITH A
NETWORK RESOURCE AND CREATING
CLIENT PROFILES AND RESOURCE
DATABASE**

FIELD OF THE INVENTION

The present invention relates to a method and apparatus for monitoring client use of and interaction with a resource downloaded from a server on a computer network, for storing monitored data, for creating a database including profiles indexed by user and/or resource identity, and for generating customized resources based upon client profiles.

BACKGROUND OF THE INVENTION

The development of software packages designed to permit simplified graphical user interface (GUI)-based access to the wealth of electronic information available over the Internet and the World Wide Web has led to a dramatic increase in the amount of information that is currently available over public computer networks. Unlike the highly controlled atmosphere of a private computer network, however, it is difficult to monitor user interaction with network resources on public networks. As a result, it is difficult for individual servers on a public network to determine how long individual users have interacted with their resources, or how much information has been downloaded. It is equally difficult for individual servers to target specialized information to a particular audience or to learn the identity of individual users on a public network.

The techniques utilized in many private networks for monitoring client use and interaction do not lend themselves to public networks. For example, user access to a server in private networks is generally obtained through the use of a unique identification number provided by the server. Details of individual user interaction with the network are closely monitored by server-resident processes, and historic databases are automatically generated and continually updated to track the nature and amount of information accessed by individual users, as well as their connection time. This information is generally used, for example, to maintain a subscriber-indexed billing database.

In a public computer network, however, use of server-resident monitoring techniques may be severely limited. In some public networks, subscribers are given unlimited access, via a service provider, to a virtually unlimited number of servers, and no permanent connection is usually made between these servers and a client machine. The nature and amount of information downloaded by individual users is not easily monitored for each client machine and only limited information concerning individual user interaction with the network may generally be captured by a server (i.e., so-called network ID and client ID).

Due largely to the lack of advanced monitoring techniques available to individual servers on a public network, the same information is generally served out to all clients on a completely untargeted basis. In other words, the same information is generally downloaded to all users that access a particular resource on a server, irrespective of individual user interests. There is therefore a need to provide servers on a public network with the ability to automatically monitor use of and interaction with resources downloaded by users so as to facilitate the targeted serving of information.

While various methods are known for obtaining information concerning user preferences, no such methods are automatic. For instance, one application, known as a "cus-

tomizable home page", permits users, upon the request of a server, to make certain choices. When a user who has done so contacts that server at a later date, the server assembles information for downloading to the user in accordance with the previously-selected choices. More specifically, the user visits a so-called "Web page" of a particular server where he or she is asked to fill in a blank form by selecting various preferences, such as links to favorite Web sites, interests in entertainment, sports, and the like. The user then submits this information to the server by clicking the so-called "submit" button of the fill-in form, which causes the client to transmit the information to the server. The server returns a Web page with a response header which creates, or "sets" an ID field located in a file on the client computer (this file is known as the "client ID" or "cookie") to include information about the user's preferences. When the user later returns to a specified Uniform Resource Locator, or "URL", on the same server, the "client ID" or "cookie" with the previously-set preference information is transmitted in the HTTP request header to the server, which can then return a Web page that is assembled according to the user-specific information. This application is disclosed, for example, in A. Gundavaram, *CGI Programming on the World Wide Web*, O'Reilly Press, 1996.

While the "customizable home page" facilitates the serving of information on a limited targeted basis, it does not provide for the automatic determination of user interests, and inconveniences the user by requesting that he or she specify various preferences. Moreover, use of a customizable home page is limited to individual Web sites and can not be "spread out" over multiple resources on different servers. In other words, while a customizable home page may be of use with respect to the particular resources located on a single server, it does not serve any purpose for other servers on a public network. A variation of this technique is used by some servers to download executable programs. For instance, one such application disclosed by G. Cornell and C. S. Horstmann, in *Core Java*, The SunSoft Press, 1996, involves the generation of "order forms" on client computers. In this application, the client machine loads a Web page from a server which has an embedded link to an executable program that downloads to and executes on the client machine. Upon execution in the client machine, the program contacts the server and retrieves a list of goods and associated prices. The program allows the user to order various goods and requires the user to fill out a form for billing purposes. The user "clicks" on the submit button of the fill-in form to transmit the information to the server. Like the customizable home page, this method of user-specific data acquisition requires the active participation of the user, and does not provide for the automatic determination of user preferences and interests.

In addition to the inability to serve out information on a targeted basis, which is of enormous concern from a marketing standpoint, the limited monitoring capabilities available to individual servers makes it difficult for servers and administrators to determine how long users have viewed their resources and how much information has been downloaded by individual users so as to be able to bill client use and interaction with network resources and to analyze the value and effectiveness of such resources. As a result, much of the information provided by a server over a public network is the same for all clients. In addition, while it is currently possible to track a user's links within the same resource, there is no standard way to track user's links across multiple resources on different servers. For example, a common occurrence in public networks is when a user is

SUMMARY OF THE INVENTION

viewing a first resource and "clicks on" a link to a second resource located on a different server. In such instances, the second resource is downloaded and the first resource is either discarded or held in background. However, there is generally no uniform way in which to monitor such occurrences. In addition, while it is currently possible to track the number of times a particular resource has been accessed, it has generally not been possible to track the length of time a particular resource has been viewed by a particular user. There is also a great deal of other valuable information concerning user interaction with a resource which would be useful to administrators, advertisers, marketing professionals and the like, but which can not be conveniently collected using current monitoring techniques.

For example, one of the largest public networks, the "Internet", has become an extremely popular advertising tool. Many companies have their own Internet "Web sites" and have also purchased advertising space within more popular Web sites of other companies. For instance, many advertisers purchase so-called "advertising banner" (or "ad banner") space within the Web page of a popular site, thereby allowing consumers to "click-through" (i.e., specify a link) to the Web site of the advertiser. In many cases, the use of an ad banner substantially increases the advertiser's exposure. Using the limited monitoring techniques available to Internet servers, however, it is difficult to determine the effectiveness of individual Web sites and ad banners. For instance, known monitoring techniques are generally limited to determining the number of times a Web page was downloaded. Similar techniques are used to determine the number of times an ad banner (which is embedded inside a Web page) has been displayed, and how many times the banner was "clicked" on to visit the Web site of the advertiser.

Generally, an ad banner is embedded inside a Web page located on a first server through the use of the known HTML tag. When a client machine passes a TCP/IP request for the Web page to the first server, the Web page is downloaded to the client, including the ad banner embedded using the tag. The tag is used to reference a resource (i.e., the "ad banner") stored on the same or a different server which captures the user's ID (via the HTTP request header) and dynamically returns an ad related image to the client for display within the Web page. At the same time, a counter representing the number of times the specific ad has been displayed is incremented. The ad banner itself may have an embedded address referring to yet another Web resource. In such an instance, if the user "clicks" on the ad banner, the client may load a resource on the second server which once again captures the user's ID and forwards the user to a Web resource which is appropriate for the displayed ad (for example, a page on the advertiser's Web site). At the same time, a counter representing the number of times the specific ad was clicked on to go to the advertiser's Web site is incremented.

While Web sites and ad banners have, in some cases, been valuable marketing tools, the limited monitoring capabilities available to servers on networks in which no permanent connection is made between a server and a client (such as the Internet) has prevented these marketing tools from being used to their full potential. Since HTTP or Web servers cannot automatically determine the amount of time and the frequency at which particular users interact with their resources, Web site administrators and advertisers cannot accurately determine the effectiveness of their resources. Since servers cannot automatically monitor user interaction and automatically obtain user preferences and interests, servers cannot assemble and serve resources targeted to individual user interests.

In view of the foregoing shortcomings of the prior art, an object of the present invention is to provide a method for tracking the use and interaction of a user with a resource downloaded from a server on a network by use of a tracking program embedded in the resource and executable by a client. Another object of the present invention is to transmit the tracking information from a client to another computer connected to the network for storage and analysis.

Still another object of the present invention is to create a database of server resources including, but not limited to, the number of times a resource has been displayed by clients, the amount of time displayed, and the type and amount of information that was displayed or transferred. This information could be used by network administrators or servers to analyze the effectiveness of the resources made available on their network servers.

Yet another object of the present invention is to provide means for creating a database of user profiles for use by advertisers and/or marketers to determine the effectiveness and value of network-based advertisements and/or marketing resources.

Still yet another object of the present invention is to provide means for creating a database of user profiles containing details of individual user interaction with and use of network resources including, for example, Network IDs (known as "IP address") and client IDs (known as "cookies") that have accessed particular resources, the amount of time spent by users interacting with and/or using particular resources, and details of choices created by individual users within a particular resource.

It is still yet another object of the present invention to provide means for assembling a resource, such as a Web page or a highly targeted ad banner, in accordance with a historic user profile.

In order to achieve the above-described and other objects and advantages, a tracking program is embedded in a file which is downloaded from a server to a client. The tracking program need not originate from the same server that sent the file, and may be obtained, for example, via an embedded URL that points to a different server. The tracking program may be part of a larger program that performs other operations (such as displaying animations, playing sounds, etc.). The tracking program is downloaded from a server and runs on the client to monitor various indicia, such as elapsed time, mouse events, keyboard events, and the like, in order to track the user's interaction with and use of the file or to monitor choices (such as selections or links to other resources or files) made by the user while within the file. The tracking program may also monitor the amount of data downloaded by the client. Operation of the tracking program commences after the program is downloaded and any required initialization occurs.

After monitoring the user's interaction with and use of the file downloaded from the server, the tracking program then automatically sends the information acquired from the client back to a server for storage and analysis. The information may be sent before or as the client exits the file, or may be sent in response to a predetermined user action. The information preferably includes any available client or network IDs.

The acquired information is preferably stored on a server and used to build historical profiles of individual users, to serve out highly targeted information based upon user profiles, as well as to extract information about how much

data was downloaded by a respective client, and how long or how often specific files were displayed or in use by the client.

Preferably, the tracking program is implemented in a network based upon the client/server model, and may be implemented in a public network such as the Internet or World Wide Web. The tracking program may monitor use of and interaction with any of the resources downloaded from a server, including an executable program, a database file, an interactive game, a multimedia application, and the like. In the case of the Internet, for example, the tracked resource may, for example, be a file such as a Web page or part of a Web page (such as an ad banner).

In one embodiment of the present invention, the tracking program is embedded in an HTML document (such as a Web site, a Web page, or part of a Web page—e.g. an “ad banner”). A TCP/IP connection is used by a client to pass a request for the HTML document. The HTML document is stored in a server running an HTTP service and contains text and one or more first embedded URLs for pointing to one or more graphical images located on a server, the images being embedded inside the HTML document using an HTML tag to specify the source URL for an image. The HTML document also contains a second embedded URL for pointing to a first executable program that runs on a server, the first executable program being embedded inside the HTML document using an HTML tag to specify the source URL for the program. A second executable program is also embedded in the HTML document by using a third URL for pointing to the second executable program. Unlike the first executable program, the second executable program is downloaded and runs on the client. The second executable program is embedded using the proper HTML tag to indicate that it is a program that is executable on the client.

After the HTML document is downloaded to the client, the graphical images are fetched using a TCP/IP connection to server resources specified by the one or more first URLs. In attempting to fetch the resource associated with the first executable program, the client causes the program to run on the server specified by the second URL. Upon execution of the first executable program, the server captures identifying indicia from the client, such as any network or client IDs resident in the HTTP request header sent by the client. The server stores this information in a client profile database.

The client also fetches the second executable program, which is the tracking program. The tracking program downloads to the client, and, after performing any required initialization, determines the current time. The tracking program also determines the current time upon the performance of a predetermined operation on the client computer by a user, such as leaving the HTML document. After calculating the amount of time the user interacted with and displayed the HTML document, i.e., by determining the difference in time values, the tracking program uploads the calculated value to the server for storage in the user profile database.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of a computer network in which the present invention may be implemented;

FIG. 2 is a block diagram of a client computer which is used in connection with various preferred embodiments of the present invention;

FIG. 3 is a flowchart diagram of a first embodiment of the present invention, which is a method for monitoring the amount of time a Web page is displayed on a client computer;

FIG. 4 is a flowchart diagram of a second embodiment of the present invention, which is a method for monitoring the amount of time a Web page is displayed on a client computer;

FIG. 5 is a flowchart diagram of a third embodiment of the present invention;

FIG. 6 is a flowchart diagram of a fourth embodiment of the present invention; and

FIG. 7 is a flowchart diagram of a fifth embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The teachings of the present invention are applicable to many different types of computer networks and may also be used, for instance, in conjunction with direct on-line connections to databases. As will be appreciated by those of ordinary skill in the art, while the following discussion sets forth various preferred implementations of the method and system of the present invention, these implementations are not intended to be restrictive of the appended claims, nor are they intended to imply that the claimed invention has limited applicability to one type of computer network. In this regard, the teachings of the present invention are equally applicable for use in local area networks of all types, wide area networks, private networks, on-line subscription services, on-line database services, private networks, and public networks including the Internet and the World Wide Web. While the principles underlying the Internet and the World Wide Web are described in some detail hereinbelow in connection with various aspects of the present invention, this discussion is provided for descriptive purposes only and is not intended to imply any limiting aspects to the broadly claimed methods and systems of the present invention.

The present invention, although equally applicable to public and private computer networks, is particularly useful for performing monitoring functions in connection with public networks which could not heretofore be performed. For this reason, implementation of the present invention will be discussed in detail in connection with the Internet and the World Wide Web. This discussion is equally applicable to any network based upon the client/server model.

Accordingly, as will be appreciated by those of ordinary skill in the art, as used herein, the term “client” refers to a client computer (or machine) on a network, or to a process, such as a Web browser, which runs on a client computer in order to facilitate network connectivity and communications. Thus, for example, a “client machine” can store and one or more “client processes.” The term “user” is used to broadly refer to one or more persons that use a particular client machine.

FIG. 1 illustrates a known computer network based on the client-server model, such as the Internet. The network comprises one or more “servers” 10 which are accessible by “clients” 12, such as personal computers, which, in the case of the Internet, is provided through a private access provider 14 (such as Digital Telemedia in New York City) or an on-line service provider 16 (such as America On-Line, Prodigy, CompuServe, the Microsoft Network, and the like). Each of the clients 12 may run a “Web browser”, which is a known software tool used to access the Web via a connection obtained through an Internet access provider. The servers allow access to various network resources. In the Internet, for example, a Web server 10 allows access to so-called “Web sites” which comprise resources in various different formats. A location of a resource on a server is identified by a so-called Uniform Resource Locator, or URL.

The "World Wide Web" ("Web") is that collection of servers on the Internet that utilize the Hypertext Transfer Protocol (HTTP). HTTP is a known application protocol that provides users access to resources (which can be information in different formats such as text, graphics, images, sound, video, Hypertext Markup Language—"HTML" etc., as well as programs). HTML is a standard page description language which provides basic document formatting and allows the developer to specify "links" to other servers and files. Links are specified via a Uniform Resource Locator or "URL". Upon specification of a link, the client makes a TCP/IP request to the server and receives information that was specified in that URL (for example another "Web page" that was formatted according to HTML) in return. The information returned may be generated in whole or in part by a program that executes on the server. Such programs are typically known as CGI (Common-Gateway-Interface) scripts and can be written using known programming languages or methods that the server supports, such as PERL or C++. A typical Web page is an HTML document with text, "links" that a user may activate (e.g. "click on"), as well as embedded URLs pointing to resources (such as images, video or sound) that the client must fetch to fully render the Web Page in a browser. These resources may not be located on the same server that the HTML document was sent from. Furthermore, HTTP allows for the transmission of certain information from the client to a server. This information can be embedded within the URL, can be contained in the HTTP header fields, or can be posted directly to the server using known HTTP methods.

FIG. 2 is a block diagram of a representative "client" computer. The same or similar computer can also be used for each of the servers. The system unit 21 includes a system bus 31 to which various components are coupled and by which communication between the various components is accomplished. The microprocessor 32 is connected to the system bus 31 and is supported by a read only memory (ROM) 33 and random access memory (RAM) 34. The ROM 33 contains, among other code, the basic input-output system (BIOS) which controls basic hardware operations such as the interaction and the disk drives and the keyboard. The RAM 34 is the main memory into which the operating system 60 and application programs, such as a Web browser 62, are loaded and cached 63. The memory management chip 35 is connected to the system bus 31 and controls direct memory access operations, including passing data between the RAM 34 and the hard disk drive 36 and the floppy disk drive 37. The CD ROM 42, also coupled to the system bus, 31, is used to store a large amount of data, e.g., multimedia programs or large databases.

Also connected to the system bus 31 are various I/O controllers: the keyboard controller 38, the mouse controller 39, the video controller 40, and the audio controller 41. The keyboard controller 38 provides the hardware interface for the keyboard 22, the controller 39 provides the hardware interface for the mouse (or other hand-operated input implement) 23, the video controller 40 provides the hardware interface for the display 24, and the audio controller 41 is the hardware interface for the multimedia speakers 25a and 25b. A modem 50 (or network card) enables communication over a network 56 to other computers over the computer network. The operating system 60 of the computer may be Macintosh OS, OS/2, AIX, BE OS or any other known operating system, and each client computer is sometimes referred to as a "client machine", a client "computer", or simply as a "client."

As noted above, the Internet includes a public network using the Internet Protocol (TCP/IP) and includes servers 10

which are accessible by clients 12. When a Web browser 62 is used to access a file on a server 10, the server 10 may send information including graphics, instruction sets, sound and video files in addition to HTML documents (Web pages) to the requesting client.

In accordance with the present invention, a tracking program is embedded in a resource, such as an HTML document which is sent from a server to a client based on a TCP/IP request. The tracking program may originate on a different server than the resource, in which case it may be obtained by the client through a TCP/IP request to the other server. The tracking program executes on a client machine, and is stored, for example, in RAM. The tracking program may monitor various indicia, such as time, mouse events, keyboard events, and the like, in order to track a user's interaction with the Web page. Thus, the tracking program may simply monitor the amount of time the user spends interacting with the Web page, or may monitor details of choices (such as links) made by individual users within a particular Web page.

In some cases, clients will "cache" a resource obtained over the network (or temporarily store a copy of the resource on the user's computer), and may use the cached copy of the resource instead of obtaining it over the Internet when the resource is needed at a later time (for example, in order to completely render a Web page). In such cases, neither the basic operations nor functions of the tracking program nor the transmission of tracked information to a server, differ from the cases where cached copies were not used.

In one embodiment of the present invention, a tracking program is embedded in an HTML of a Web page and downloaded by a client. The tracking program may monitor operation of a peripheral input device connected to the client machine, such as a keyboard or mouse, keep a record of which choices, if any, are made by a user, and may monitor the length of time the user has displayed the Web page in addition to the time spent interacting with a particular part of it. While in the preferred embodiment, the tracking program is embedded in an HTML document, those skilled in the art will recognize that other mechanisms are possible for embedding the tracking program in the client hardware, and the patent is not limited to implementation as an executable program embedded in an HTML document. For example, the tracking program may be downloaded and installed in a client process, as would be the case for a so-called "plug-in" or "helper" application. Alternatively, the tracking program can be built into a client application or client process such that it need not be separately downloaded and installed. In addition, the teachings of the present invention are not limited to use on the Internet or the World Wide Web. For instance, the tracking program of the present invention may be utilized on a so-called "Intranet".

As noted above, a client process, such as a Web browser running on the client machine, uses a TCP/IP connection to pass a request to a Web server running an HTTP service (or "daemon" under the UNIX operating system). The HTTP service then responds to the request, typically by sending a Web page formatted in the Hypertext Markup Language, or HTML, to the browser. The browser displays the Web page using local resources (e.g., fonts and colors). Unless the tracking program is already resident in the client, it is embedded in the Web page and downloaded to the client along with the Web page. The tracking program is executed after any required initialization has occurred. The tracking program may monitor the length of time the user remains in the Web page, or any one or more portions thereof, and may track some or all mouse and keyboard events to provide

meaningful data to the server concerning the user's interaction with the Web page.

In its simplest form, the tracking program is a timer program linked to an HTML document and is downloaded and executed on a client when the HTML document is served to the client in response to a client TCP/IP request. During or after the client formats and displays the Web page specified by the HTML document, the tracking program begins a software timer to monitor the amount of time the Web page is displayed on the client computer.

When the user leaves the Web page (for example, by exiting the Web page or "clicking" on a link to another resource on the same or another server), the tracking program sends the monitored time to another computer on the Internet for storage and analysis.

As illustrated, for example, in FIG. 3, the client issues a TCP/IP request for a Web page located on a Server A (S301). After a handshaking period, the Server A begins to send the HTML formatted document, which contains an embedded URL referencing the tracking program. The client additionally issues a TCP/IP request to the Server B referenced by the embedded URL in order to obtain the tracking program (S302). The client also makes any other TCP/IP requests (S303) to obtain any other resources (such as images, video or sound) needed in order to fully render the Web Page (S304). Each of such resources are typically referenced by individual URLs embedded in the HTML document. These requests need not occur in any specific order and may reference resources located on any server. In addition, the information requested may be received in any order. When the tracking program has been obtained, the client process (i.e., the Web browser) saves the tracking program to RAM (S305). After any necessary initialization, the tracking program initiates a software timer to monitor the amount of time the Web page is displayed (S306). When the client leaves the Web page (S307), the tracking program calculates the amount of time the user has interacted with and displayed the Web page and sends this information to a server. Other available client information, such as the network ID and client ID, or so-called "Cookie" of the client, is also sent to the server (S308). If desired, other information concerning the client computer may be automatically acquired and sent to the server, such as the type of hardware in the client computer and various resources that are resident on the client computer.

Due to the technical limitations imposed by the Internet, the JAVA programming language was applied to the Internet in 1995 by programmers at Sun Microsystems, Inc. of Mountain View, Calif. For example, some of the fundamental technology issues facing network programmers and engineers are portability, bandwidth and security. Portability allows the same executable code to run across multiple operating systems. Bandwidth specifies the amount of information that can transfer across the network at any time. For instance, high-speed lines categorized as T1 through T3 can transmit data at 1.544 through 45 megabits per second, ISDN lines can transmit data at rates of 64 through 128 kilobits per second, and standard phone lines, over which most users transmit data, currently transmit using modems at approximately 28.8 kilobits per second. In the case of a 640x480 pixel window on a computer display that is capable of displaying images in 256 colors (which requires one byte per pixel), in order to display the window's contents requires 307,200 bytes of data. To create an animation, programs typically display 15 to 30 different images per second. Given a 640x480 window, 15 to 30 frames per second would require 4,608,000 to 9,216,000 bytes per second. Because

many users are currently browsing the Web using 28.8 kilobit (or slower) modems, there is simply not enough bandwidth to download animation screens. As a result, many Web sites today resemble magazines whose images are for the most part static (unchanging). However, to satisfy an audience that spends many hours in front of dynamic television images, Internet programmers and engineers must provide a way to animate Web sites. One solution is to download programs written in the JAVA programming language that implement the animation.

Animation is only one example of the use of JAVA. Using JAVA, programmers can create stand alone programs similar to those that programmers can develop using C++, and can also create so-called "applets" that run within a Web browser. To address security issues, JAVA developers ensured that a programmer could not develop a computer virus using a JAVA applet and that an applet could not arbitrarily transfer information concerning a user's system (such as a file on the user's system) back to the server. Thus, JAVA applets have limited operations. For example, a JAVA applet generally cannot currently read or write files on the user's system. In this way, an applet cannot store a virus on a user's disk or arbitrarily read information stored on a user's disk. In addition, for other security and stability reasons, JAVA developers eliminated or changed many features of the C and C++ programming languages, such as pointers, with which advanced programmers could bypass JAVA's security mechanisms.

JAVA applets run within a "JAVA-enabled client", such as Netscape Navigator version 2.0 (Windows 95 or Windows NT versions only) or later, or Microsoft's Internet Explorer version 3.0, or later. In addition, since most users browse with personal computers running Windows, Macintosh, UNIX-based systems, and the like, the JAVA developers designed JAVA to be portable, or "platform-independent". Thus, the same JAVA applets can be downloaded and run in any JAVA-enabled client process, irrespective of the platform type.

JAVA applets can be used by developers to create sophisticated, fully interactive multimedia Web pages and Web sites executable on any JAVA-enabled client. Representative JAVA applets are disclosed, for example, by O. Davis, T. McGinn, and A. Bhatani, in *Instant Java Applets*, Ziff-Davis Press, 1996.

Since JAVA provides the ability to download complex programming instructions in the form of applets that are executable by a JAVA-enabled Web browser, the tracking program of the present invention may be implemented in the JAVA programming language. As will be readily appreciated by those of ordinary skill in the art, however, the teachings of the present invention are not limited to JAVA applets or to the JAVA programming language whatsoever. In connection with the Internet, for example, the present invention may also be implemented in a so-called "Active-X" environment, in which the tracking program is written as an Active-X component.

As will be further appreciated by those of ordinary skill in the art, security restrictions may, in some cases, prevent one from having direct access to information stored on a client's hard disk, such as client IDs. In such cases, other means may be used to obtain this information. For example, when a Web browser makes a request for information from a server it typically includes certain information about the client in the "HTTP request header." The server receiving the request can obtain and store this information using known means implemented, for example, in a so-called "CGI script"

executable on the server. Therefore, one way of obtaining client identifying indicia is to embed a request in the HTML file for another resource on a server that will obtain and store the indicia. This resource may be a program (such as a CGI script) that captures relevant information and stores it. This information can then be combined with information monitored by the tracking program to provide a more detailed knowledge base. This embedded request may be in addition to the embedded tracking program. Representative CGI scripts capable of capturing client identifying indicia are disclosed by A. Gundavaram, in *CGI Programming on the World Wide Web*, O'Reilly Press, 1996.

In order to store client-identifying indicia, such as a user's network ID (IP) and client ID numbers (cookies) and associated tracking information, a database is set up on a server. This may be done in any known manner, such as by using a commercially-available database program designed, for example, for the high-speed processing of large databases. In the case of the tracking program described above, the information stored in the server database may include the network ID, client ID, the associated link (the URL of the Web page), the amount of time the user spent interacting with the Web page, and any selections or choices made by the user while interacting with the Web page. Thus, the above-described tracking program permits Web site administrators and Internet advertisers, for example, to determine not only the number of user visits or hits made to a particular Web page, but also permits the accurate determination of the length of time users have displayed and/or interacted with their Web page. This is invaluable information to Internet advertisers, among others, and permits advertisers to make informed decisions as to the effectiveness and value of particular Web pages and/or ad banners.

A more particular embodiment of this aspect of the invention is illustrated in FIG. 4. A Web page (or HTML document) is requested by the client from a first server A, using TCP/IP and HTTP protocols (S401). This HTML document contains text, as well as embedded URLs that point to graphical images (e.g. GIF format image files) also located on the first server A. The images, in general, may be located on any HTTP server on the Internet. These images are embedded inside the Web page using the known HTML tag, which allows one to specify the source URL for an image, as well as additional information such as size and other layout parameters. These images will then be fetched by the client using TCP/IP and HTTP protocols from Server A (S402) and rendered on the browser (S405). The Web page (or other Web or HTML document) additionally includes embedded URLs which point to two resources that reside on a second server "B". One of the resources is an executable program, which executes on Server B, and is a CGI script. This resource is also embedded inside the Web page using the tag. Thus, in attempting to render the Web page, the client will automatically fetch this resource (S403), which forces execution of the CGI script on the second Server B and the return of information output from the script to the client. In this case, the information returned to the client is formatted as an GIF image type which is extremely small as well as completely transparent (S403B). When the CGI script executes, it may collect information from the HTTP request header such as browser type, network ID (IP address), and if set, client ID ("cookie"), as well as any additional available information such as time of execution and the URL of the Web page, and store it in a database—for example using SQL (S403A, S404). In step S403B, the CGI script returns information to the client, which includes a response header which indicates (among other information),

that the return type is an image, that this resource should not be cached by the client, and if no client ID is set and the client supports it, that a client ID is to be set to a value generated by the script.

In addition, the CGI script may monitor the number of times the Web page has been accessed in general. On the other hand, another CGI script located on the same or another server may be used for this purpose. This process may be carried out by simply incrementing a counter each time the resource is accessed, or may be conducted at any other time by merely counting the number of entries made in a stored record of requests made for the resource.

The other resource located on Server B is a JAVA applet, the tracking program. This resource can also be located on any other server, and is embedded in the Web page using the known HTML <APPLET> tag, which allows one to specify the source URL (through the CODE and CODEBASE parameters) as well as additional size, layout and initialization parameters. The client, in attempting to render the Web page, will automatically fetch the applet by making a request to Server B using the TCP/IP and HTTP protocols (S406). Soon after it has received the JAVA code for the tracking program, it will first execute the INIT (initialization) method of the applet (S407) and then the START method. The START method will make note of the current time using standard JAVA methods (S408). The STOP method of the applet which is executed, for example, when the user leaves the Web page (S409), will compute the difference between the current time and the time noted during execution of the START method. This difference, which is the time between execution of the STOP and execution of the START methods, is sent to the Server B for storage and analysis (S410). The information can be sent using standard JAVA network methods, such as opening a URL connection to a second CGI script on Server B (or any other server) designed to capture the tracked information (S410A). This second CGI script can then obtain any information tracked and transmitted by the applet as well as any available information in the HTTP request header. This information can be stored in a database on Server B or elsewhere. If necessary, the information stored by both scripts may be combined into one or more complete databases. As will be understood by those of ordinary skill in the art, acquisition of information by the server need not be conducted using CGI scripts. For instance, this information may be acquired by any other server-resident process designed for this purpose, or may be uploaded by the tracking program or other client-resident process, such as by a direct connection to a resource located on a server (i.e., a database), or by using any other known process.

The database thus constructed can be indexed by resource identity and may contain information about users who have visited the Web page, such as their network and client IDs, how often they visited the Web page, how long the Web page was displayed, and so on. Additionally, if the above-mentioned tracking mechanism is implemented across various Web pages in a particular Web site, the database thus constructed may contain similar information about the different Web pages in the Web site. Similarly, the information acquired by the tracking program may be combined with a process for monitoring the number of times the Web resource has been accessed. An analysis of the data on a user-indexed basis would facilitate the determination of individual user interests and the like. On the other hand, analysis of the data on a resource-indexed basis would allow the determination of, for example, which Web pages are viewed the longest and/or most often either by users in

general, or by specific users. Thus, it would be possible to determine if there were different types of users that preferred different sections of the Web site (because, for example, they spent more time browsing different sections of the Web site). Additionally, if the above-mentioned tracking program is attached to an ad banner that is embedded in multiple Web pages across different Web sites (as is typically the case with ad banners), the database thus constructed may contain information about how often and for how long the different pages that contained the ad banner were displayed, as well as more specific information about users that visited those pages. With this information, advertisers could determine the accuracy of data supplied to them by Web site administrators about the number of times their ad banner was displayed, as well as learn how long the Web page containing the ad banner was displayed—a number that would be of great use in determining the effectiveness of their advertising.

In another embodiment, the software timer of the tracking program may be initiated or stopped when the user incurs a keyboard or mouse event, such as by “clicking” on a specified area of an ad banner. This is illustrated in the flowchart shown in FIG. 5. Operation of the system in this embodiment is similar to that shown in FIG. 3. Thus, the client first issues a TCP/IP request (S501). After a handshaking period, a first Server A begins to send an HTML formatted document, which contains an embedded URL referencing the tracking program. The client additionally issues a TCP/IP request to a second Server B referenced by the embedded URL in order to obtain the tracking program (S502B). The client also makes any other TCP/IP requests to obtain any other resources (such as images, video or sound) needed in order to fully render the Web Page (S502A). Each of such resources are typically referenced by individual URLs embedded in the HTML document. These requests need not occur in any specific order, and the information requested may be received in any order. When the tracking program has been obtained, the client process (i.e. the Web browser) saves the tracking program to RAM (S503B). In this case, the tracking program commences a software timer upon the detection of a predetermined user action (S504). When the user performs another predetermined action (S505), the tracking program calculates the amount of time between the predetermined user actions, and sends this information, along with other available client information, to the server (S506).

Thus, for instance, the software timer of the tracking program may be used in monitor the amount of time a user spends interacting with a portion of a Web page. For example, if the Web page is provided with an interactive resource such as a game or an information resource activated by clicking on a particular button, the tracking program may determine how long a user has interacted with such a selection. In the case of a Web page provided with an ad banner, the tracking program can be designed to monitor the amount of time a user has interacted with the ad banner.

The tracking program may be used not only to monitor the time spent by a user in a Web page or an ad banner, but may also be used to create a more complex “historical” user profile to permit the server to assemble a Web page or target an ad banner based upon the diverse interests of respective users.

For example, when a user is exposed to an ad banner having information targeted to their particular interests, the user is more likely to interact with that ad banner for a longer period of time and on a more frequent basis, thereby increasing the value of that ad banner. In accordance with

the present invention, in order to learn the particular interests of respective users, an ad banner may include specific information permitting the user to interact in different ways with the banner. The ad banner may have pull-down menu options, clickable buttons or “hot-spots”, keyboard input, or any number of input mechanisms, whose selection or action upon in a designated manner causes corresponding events to take place in the ad banner such as the generation or synthesis of sounds, the display of images, video, or graphic animations, or the presentation of different types of information to the user, perhaps with additional choices. Such information may, for example, include links to interactive games, links to entertainment information, sports-related games and/or trivia, and the like, or information concerning particular goods and services, or means by which to order or purchase specific goods and services. The more choices that are made available, the more information that can be acquired concerning the user’s particular interests. Of course, an unlimited number of possibilities are available, depending upon the application, and an exhaustive listing of such possibilities cannot be provided herein.

In this case, the tracking program is downloaded, as described above, with the HTML document in response to a TCP/IP client request. As above, the tracking program may monitor the amount of time the user spends displaying both the Web page and the ad banner embedded in the Web page as a whole, but also monitors the user’s interaction with the Web page and the ad banner, such as by monitoring each of the choices made by the user within the Web page and ad banner. Thus, for example, if an interactive sports-related game is included in the Web page, the tracking program will determine if a user has played the game, what his or her score was, how long they played the game, and any other possible information. If a choice of different games, each directed to a different interest, are made available to users within the same ad banner, it is possible to determine what is of most interest to the user by the selection of the game. In addition, the ad banner may be provided with multiple links to other, diverse Web sites, such as Web sites relating to sports, entertainment, general information, technology, history, and the like. The tracking program monitors which of the various links are selected and provides this information to the server. As discussed above, other available client information may also be sent to the server. This information is sorted and stored in the server database and may be analyzed manually or automatically.

The tracked information may be used to assemble resources geared toward the user’s interests. Based upon the historic user profiles created in the server database, downloading of information to the same client on a subsequent visit to the same or different Web page may be done on a more intelligent basis. For example, users who have previously expressed an interest in sports-related trivia (as indicated by their previously tracked behavior) may be served with information targeted to audiences interested in sports. Similarly, users who have expressed greater interest in technology may be served with technology-related information that would be of much less interest to other users. The assembly of a resource such as a Web page may be easily accomplished. For example, the HTML document of the Web page may include a plurality of embedded resources. Previous choices made by a user on a particular client computer and stored in a user profile database may be used to determine which of the resources is to be downloaded to that client using simple logical processing instructions. For instance, a user profile which indicates that a user has a greater interest in sports-related information than in histori-

cal information may be used to download sports-related resources, such as GIF-type images and advertisements. Since the user has previously expressed a greater interest in sports, sports-related advertisements may therefore be targeted to that user.

A particular implementation of this mechanism is illustrated in FIG. 6. A Web page is requested by the client from Server A (S601). This Web page contains text, as well as embedded images which must be fetched from Server A (S602) and rendered (S605). In addition, the Web page contains embedded URLs that point to two resources on Server B. The first resource is a first CGI script 1, which is embedded inside the Web page using the standard HTML tag (S603). In attempting to render the Web page, the client will automatically fetch the resource associated with the tag on Server B, which will result in execution of the CGI script 1. This CGI script 1 can capture client information such as Network ID or Client ID (S603A). The CGI script also returns a transparent image (S603B).

The other resource on Server B is a Java applet, which is a combination ad banner and tracking program. This may be stored on any server. In attempting to render the Web page, the client will automatically fetch the Java code (S604), download, initialize, and start operation of the applet (S607, S608). After the applet is initialized, it contacts Server B to obtain other resources it needs in order to display images, play sounds, or control its overall look and behavior. In fact, the applet may obtain these resources by executing one or more CGI scripts or other processes that reside on Server B or elsewhere (S607). Based on information provided to these scripts through standard HTTP methods, including client information (S607A), such as network and client IDs, any other information such as the URL of the Web page, as well as information captured by the CGI script 1, and the previously constructed historical database profile (S607B), different information (images, sounds, text, etc.) may be returned to the applet. Such information can therefore be selected by the scripts based on Network and/or Client ID, the URL of the Web page, and the previously constructed client profile. This may be accomplished in the manner described above.

The STOP method of the applet which is executed, for example, when the user leaves the Web page (S609), will compute the difference between the current time and the time noted during execution of the START method. This difference, which is the time between execution of the STOP and execution of the START methods, is sent to the Server B for storage and analysis (S610). The information can be sent using standard JAVA network methods, such as opening a URL connection to a second CGI script on Server B designed to capture the tracked information (S610A, S610B). In step S610A, the second CGI script may obtain any information acquired by the tracking program (i.e., the JAVA applet), as well as client identifying indicia transmitted by the client, such as in the HTTP request header. This information can be stored in a database on Server B. If necessary, the information stored by both scripts may be combined into one more complete databases.

In this embodiment of the present invention, two distinct databases may be created. The first database is indexable by resource identity (such as URL), and includes information such as URL of the Web document, number of times accessed, identity of clients that accessed the Web document, amount of time displayed, amount of data displayed, average time displayed, number of times accessed, and the like. In the case of an ad banner or other embedded resource which may be accessed by a link made

by a user while browsing another resource, the database may include additional information such as "click-through rate" (the number of times the ad banner was clicked on to go to the Web site of the advertiser), and the like.

5 A second database that may be created is indexable by individual client, and includes information concerning individual client's interests and preferences. These separate databases may be combined in a single database indexable by client or resource identity.

10 In another embodiment, illustrated in FIG. 7, the tracking program is used to create a database of information about a Web site (or, if desired, across multiple Web sites on multiple servers). In this case, the same tracking program is embedded in multiple Web pages served up by the same Server A. The tracking program in general originates from a Server B (but may also originate from Server A). The tracking program will monitor the time the Web page was displayed, and may capture any other information available to it. For example, the tracking program can determine the URL of the Web page it is embedded in and may determine the amount of information downloaded by the client.

In particular, a Web page is requested by the client from Server A (S701). This Web page contains text, as well as embedded images which must be fetched from Server A (S702) and rendered (S705). In addition, the Web page contains embedded URLs that point to two resources on Server B. The first resource is a CGI script, which is embedded inside the Web page using the standard HTML tag (S703). In attempting to render the Web page, the client will automatically fetch the resource on Server B, which will result in execution of a CGI script 1. This CGI script 1 can capture client information such as Network ID or Client ID (S703A) and returns a transparent image (S703B). The other resource on Server B is a Java applet. This may be stored on any server. In attempting to render the Web page, the client will automatically fetch the JAVA code, store it in RAM, initialize, and start operation of the applet (S707). The START method of the applet is executed and the applet takes note of the current time (S708). Thereafter, the applet contacts the Server A and, if security restrictions allow it, the applet queries the Server A for the page it is embedded in, determines its size, as well as the URLs of other embedded resources (such as images or video), and requests header information about these resources in order to determine their size (S709). In this case, the tracking program may determine the size of the fully rendered Web page, (i.e., the number of bits that must be downloaded in order to fully render the Web page). If the tracking program is part of a larger embedded application that displays information downloaded from a server (such as a live news feed applet), the tracking program can also monitor the amount of information downloaded and displayed by the applet. Before or as the user leaves the Web page (S710), the tracking program can transmit this information to Server B for storage and analysis (S711, S711A, S711B). In this manner, it is possible to build a database of accurate information concerning how often different pages of a Web site are requested, how long they are displayed, and how much information was downloaded. This information would be of use to Web site administrators in order to judge the popularity of different Web pages, as well as for example to set advertising rates for any embedded advertisements.

In yet another embodiment, the tracking program is used to assemble a bill for the user's access to information. For example, users who have access to a live news or entertainment feed may be charged according to the amount of information displayed, either according to bit size or time, or

both. Imagine that the tracking program is attached to a live feed applet. The tracking program monitors the time the information is displayed and the amount of bits downloaded and automatically transmits this information back to a server when the user leaves. Together with the user's ID (client and network), and billing information that the user was previously requested to enter, it is possible to determine the correct charge for the user. Similarly, a user could be charged and billed for time spent on a Web page, as well as amount of information downloaded by him or her.

The methods embodied in the invention may be used to create web resources with so-called "persistent" state. That is, the tracking program, in addition to the client profile database, may also be used to create a Web resource that appears to automatically "remember" the user's previous interactions on the Web resource. This may be implemented as in FIG. 6. For example, consider a Web page with an embedded Crossword program which also incorporates tracking mechanisms. When the page is rendered and the Crossword program commences, a user is able to use the keyboard and mouse to fill in letters on the Web page based on clues that are displayed. At the same time, these choices are tracked, along with any other information including but not limited to time. Before or at the time the user leaves the Web page, the tracked information is sent to a server for storage (S610). When the user later returns to that page, the network or client ID is used to automatically fill in the letters in the crossword that were previously selected (As in S607-607C).

Although the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that various modifications of the invention can be practiced within the spirit and scope of the appended claims. Thus, for example, the scripts used to transfer data need not be CGI scripts but could be a dedicated server or a direct connection to the database, such as using JDBC (Java Database Connectivity) to place data into the database.

In addition, while the preferred embodiments have been described in connection with JAVA applets that are executable on a client, the tracking of user interaction may be accomplished by a client executable program written in a language other than JAVA. For example, the teachings of the present invention may be accomplished using Active-X components in conjunction with the Internet Explorer Web browser. In addition, the tracking program need not be a program that executes on the client computer. For example, the tracking program may comprise a CGI script located on a server. Upon execution of the CGI script, the time at which a Web page is downloaded may be determined. By modifying Web browser software using appropriate instructions, the browser can be used to send a signal to the server that downloaded the Web page upon the occurrence of a predetermined user operation (such as exiting the Web page or clicking on a link to another Web page or resource). In this manner, a program running on the server can be used to determine the total time period the user has interacted with and displayed the Web page.

It should also be appreciated that while the preferred embodiments of the tracking program use a single database to store the information, multiple databases could be used to store and process the information.

In addition, while in the preferred embodiments of the tracking program the server that originated the tracking program and the database reside on the same machine, this is not a requirement of the present invention. The database may instead reside on a separate machine from that which

serves the tracking program. Similarly, while in the preferred embodiments the server that originates the network resource, or Web page (Server A), and the server that originates the tracking program (Server B) are different servers, this is not a requirement of the present invention. The network resource (Web page) and the tracking program may be served out by the same server.

It should also be appreciated that while in the preferred embodiments the tracking program uses the HTTP and TCP/IP protocols, other network data transmission protocols could be used that implement the same functionality. Moreover, use of an HTML formatted Web page is not necessary. The information supplied to the user may not be in the form of an HTML or Web document such as a Web page, but can be some other form of information. In addition, the tracking program need not be downloaded to the client from the server, but can be an added module to the client application or Web browser running on the client, or may be stored elsewhere on the client machine. For example, in the former case, added modules could be plug-ins and in the latter case could be referred to as cached resources. In such cases, the client application or Web browser would include appropriate means to enable activation of the tracking program and the uploading of a client profile based upon the user's interaction with a Web page or network resource.

Moreover, although in the preferred embodiments it is envisioned that the network resource or Web page is downloaded from a remote server, this is not a limitation of the invention. The precise location of the target document or server is not important. For example, the target document may even be located on the hard drive of the client machine.

Also, while in the above-described embodiments, the client profile is created automatically using information acquired by the tracking program and one or more CGI scripts and is stored in the server database, the client profile can be created in a different manner and/or supplemented by additional information. For example, one such technique for creating a client profile is through the use of HTML "fill-in" form tags. In such cases, the client profile is created not by the tracking program, but instead by the client. Based on the client profile, the server can serve out information targeted to the client's interest, as revealed by the fill-in form.

Also, while the preferred embodiments have been described in the context of Web browser software, the techniques of the invention apply equally whether the user accesses a local area network, a wide area network, a public network, a private network, the Internet, the World Wide Web, or the like, and whether access to the network is achieved using a direct connection or an indirect connection. For example, in connection with the World Wide Web, the teachings of the present invention apply whether a network connection is obtained via a direct Internet connection or indirectly through some on-line service provider. Thus, the "computer network" in which the invention is implemented should be broadly construed to include any computer network in which one or more clients is connectable to one or more servers, including those networks based upon the client-server model in which a client can link to a "remote" document (even if that document is available on the same machine, system, or "Intranet").

It should also be appreciated that while in the preferred embodiments the tracking program is downloaded with the Web page from the server, this is not a limitation of the invention. The tracking program need not be embedded within an existing Web page, but rather may be embedded

within a Web browser or supported elsewhere within the client itself. Thus, the tracking program may be initiated whenever a call to a Web page or network resource is made, such as when a search to a particular URL is initiated, or when a previously-stored URL is launched.

We claim:

1. In a computer network having one or more servers connectable to one or more clients, a method of monitoring the amount of time a user interacts with and displays a file downloaded from a server, comprising the steps of:

using a client to specify an address of a resource located on a first server;

downloading a file corresponding to the resource from the first server in response to specification of the address;

using the client to specify an address of a first executable program located on a second server, the address of the first executable program being embedded in the file downloaded from the first server, the first executable program including a software timer for monitoring the amount of time the client spends interacting with and displaying the file downloaded from the first server;

downloading the first executable program from the second server to run on the client so as to determine the amount of time the client interacts with the file downloaded from the first server;

using a server to acquire client identifying indicia from the client; and

uploading the amount of time determined by the first executable program to a third server.

2. A method of monitoring according to claim 1; wherein the step of using a client to activate a link to a resource located on a server comprises the step of using a TCP/IP connection to pass a request for an HTML document from the client to the server.

3. A method of monitoring according to claim 2; wherein the HTML document is a Web page formatted in HTML containing text and one or more embedded URLs for pointing to a graphical image type located on a server, the image type being embedded in the HTML document using an HTML tag to specify the source URL for an image and predetermined layout parameters.

4. A method of monitoring according to claim 3; wherein the HTML document further comprises a URL pointing to a process that executes on a server and being embedded in the HTML document using an HTML tag; and the step of downloading includes the steps of attempting to fetch the resource specified by the HTML tag using the client by issuing an HTTP request having a request header, executing the process in response to the attempt to fetch by the client, capturing client-identifying indicia from the HTTP request header, and storing the client-identifying indicia in a first database.

5. A method according to claim 4; wherein the process that executes on a server comprises a CGI script.

6. A method of monitoring according to claim 3; wherein the HTML document further comprises a URL pointing to a program that executes on a server and has an address that is embedded in the HTML document; and the step of downloading includes the steps of fetching the program with the client by issuing an HTTP request having a request header, executing the program in response to fetching by the client, capturing client-identifying indicia from the HTTP request header, and storing the client-identifying indicia in a first database.

7. A method according to claim 1; wherein the first executable program comprises a software component adding

functionality to a client application and is downloaded from a server and installed in an application running on the client.

8. A method according to claim 7; wherein the software component comprises a plug-in or helper-application.

9. A method according to claim 7; wherein the software component comprises an Active-X component.

10. A method according to claim 1; wherein the step of using a server to acquire client identifying indicia from the client comprises the steps of using the client to specify an address of a second executable program located on a respective server, the address of the second executable program being embedded in the file downloaded from the first server, the second executable program including a routine for acquiring client identifying indicia in response to activation of an address thereto, and using the respective server to run the second executable program to acquire client identifying indicia from the client.

11. A method according to claim 10; wherein the second executable program is a CGI script.

12. A method according to claim 10; wherein the second executable program is a JAVA applet that is downloaded and runs on the client.

13. A method according to claim 1; wherein the step of specifying an address of the resource includes the step of using a TCP/IP connection to pass a request having an HTTP request header containing client identifying indicia to the second server.

14. A method of monitoring according to claim 13; wherein the step of acquiring client identifying indicia includes the step of using the server to acquire the client indicia from the HTTP request header.

15. A method of monitoring according to claim 1; further comprising the step of incrementing a count value corresponding to the resource in the first server in response to downloading of the file corresponding to the resource.

16. A method of monitoring according to claim 15; further comprising the step of storing the count value in a database.

17. A method of monitoring according to claim 1; wherein the step of acquiring client identifying indicia from the client comprises the steps of embedding a link to the second executable program in the file downloaded from the first server, the second executable program being executable on the third server, using the client to activate the link to the second executable program by sending a request having a request header containing client identifying indicia in an attempt to fetch the second executable program; using the third server to execute the second executable program in response to activation of the link using the server to check the request header issued by the client to determine if a client ID has been set for the client, and, if no client ID has been set, setting an ID for the client, and storing the client ID in a first database.

18. A method according to claim 17; wherein the client ID comprises a cookie.

19. A method according to claim 1; wherein the first executable program is cached on the client.

20. A method of monitoring according to claim 1; wherein the second and fourth servers comprise a single server.

21. A method of monitoring according to claim 1; wherein the second through fourth servers comprise a single server.

22. A method of monitoring according to claim 1; wherein the resource located on the first server comprises a Web document and includes an embedded URL to another resource located on the second server, the other resource comprising an ad banner.

23. A method of monitoring according to claim 1; wherein the resource located on the first server comprises a Web

document and includes an embedded URL to another resource located on a fifth server, the other resource comprising an ad banner.

24. A method of monitoring according to claim 1; wherein the step of specifying an address of a resource located on a first server includes the step of obtaining the resource using a URL.

25. A method of monitoring according to claim 1; wherein the file downloaded from the first server is an HTML document.

26. A method of monitoring according to claim 1; wherein the resource located on the first server is an HTML document, and the step of specifying an address of the resource includes the step of using a TCP/IP connection to pass a request having an HTTP request header containing client identifying indicia to the second server.

27. A method of monitoring according to claim 1; further comprising the step of storing the client identifying indicia in a first database on a server.

28. A method of monitoring according to claim 1; further comprising the step of storing the calculated amount of time in a first database on a server.

29. A method of monitoring according to claim 1; further comprising the step of storing the address of the resource located on the first server in a first database.

30. A method of monitoring according to claim 1; further comprising the steps of storing the client identifying indicia in a database, storing the calculated amount of time in a database, and storing the address of the resource located on the first server in a database.

31. A method according to claim 1; wherein the first executable program is an applet written in JAVA.

32. A method according to claim 1; wherein the software timer of the first executable program commences operation after a predetermined user operation.

33. A method according to claim 1; wherein the first executable program is downloaded in response to a predetermined user operation.

34. A method according to claim 1; further comprising the steps of providing a user of the file with one or more choices or selections requiring manual entry using an input peripheral device connected to the client, monitoring choices and selections made by the user, sending the information back to a server and storing the monitored choices and selections in a database.

35. In a computer network having one or more servers connectable to one or more clients, a method of monitoring client use and interaction with a resource located on a server, comprising the steps of:

using a server to monitor requests for the resource;
downloading a file corresponding to the resource to a client in response to a request for the resource transmitted to the server;

downloading a first executable program to the client, the address of the first executable program being embedded in the file downloaded to the client, the first executable program including a software timer for monitoring the amount of time the client spends interacting with and displaying the file;

counting the number of times the file has been downloaded; and

storing an address of the file, an amount of time the file has been interacted with and displayed by clients, and the number of times the file has been downloaded in a first database on a server.

36. A method of monitoring according to claim 35; wherein the step of requesting a resource located on a server

comprises the step of using a client to activate a link to a resource located on a server using a TCP/IP connection to pass a request for an HTML document from the client to the server.

37. A method of monitoring according to claim 36; wherein the HTML document is a Web page formatted in HTML containing text and one or more embedded URLs for pointing to a graphical image type located on a server, the image type being embedded in the HTML document using an HTML tag to specify the source URL for an image and predetermined layout parameters.

38. A method of monitoring according to claim 37; wherein the HTML document further comprises a URL pointing to a process that executes on a server and being embedded in the HTML document using an HTML tag; and the step of downloading includes the steps of attempting to fetch the resource specified by the HTML tag using the client by issuing an HTTP request having a request header, executing the process in response to the attempt to fetch by the client, capturing client-identifying indicia from the HTTP request header, and storing the client-identifying indicia in a first database.

39. A method according to claim 38; wherein the process that executes on a server comprises a CGI script.

40. A method of monitoring according to claim 38; wherein the HTML document further comprises a URL pointing to a program that executes on a server and has an address that is embedded in the HTML document; and the step of downloading includes the steps of fetching the program with the client by issuing an HTTP request having a request header, executing the program in response to fetching by the client, capturing client-identifying indicia from the HTTP request header, and storing the client-identifying indicia in a first database.

41. A method according to claim 35; wherein the first executable program comprises a software component adding functionality to a client application and is downloaded from a server and installed in an application running on the client.

42. A method according to claim 41; wherein the software component comprises a plug-in or helper-application.

43. A method according to claim 41; wherein the software component comprises an Active-X component.

44. A method according to claim 35; further comprising the steps of using the client to specify an address of a second executable program located on a respective server, the address of the second executable program being embedded in the file downloaded from the first server, the second executable program being a routine for acquiring client identifying indicia in response to activation of an address thereto, using the respective server to run the second executable program to acquire client identifying indicia from the client, and storing the client identifying indicia in a database.

45. A method according to claim 44; wherein the second executable program is a CGI script.

46. A method according to claim 44; wherein the second executable program is a JAVA applet that is downloaded and runs on the client.

47. A method according to claim 35; wherein the step of requesting the resource includes the steps of specifying an address of the resource using a TCP/IP connection to pass a request having an HTTP request header containing client identifying indicia to the second server.

48. A method of monitoring according to claim 47; wherein the step of acquiring client identifying indicia includes the step of using the server to acquire the client indicia from the HTTP request header.

49. A method of monitoring according to claim 35; further comprising the step of incrementing a count value corre-

sponding to the resource in a server in response to downloading of the file corresponding to the resource.

50. A method of monitoring according to claim 49; further comprising the step of storing the count value in a database.

51. A method of monitoring according to claim 35; further comprising the step of acquiring client identifying indicia from the client by embedding a link to a second executable program in the file downloaded from the server, the second executable program being executable on a respective server, using the client to activate the link to the second executable program by sending a request having a request header containing client identifying indicia in an attempt to fetch the second executable program, using the respective server to execute the second executable program in response to activation of the link, using the respective server to check the request header issued by the client to determine if a client ID has been set for the client, and, if no client ID has been set, setting an ID for the client, and storing the client ID in a database.

52. A method according to claim 51; wherein the client ID comprises a cookie.

53. A method according to claim 35; wherein the first and second databases comprise a single database.

54. A method according to claim 53; wherein the step of downloading a file corresponding to the resource to a client comprises the steps of assembling a file in accordance with information stored in the single database.

55. A method according to claim 35; further comprising the steps of providing a user of the file with one or more choices or selections requiring entry using an input peripheral device connected to the client, monitoring choices and selections made by the user, and storing the monitored choices and selections in a database.

56. A method according to claim 35; wherein the first executable program comprises a plug-in application that is downloaded from a server and installed in an application running on the client.

57. A method according to claim 35; wherein the first executable program is cached on the client.

58. A method of monitoring according to claim 35; wherein the resource located on the server comprises a Web document and includes an embedded URL to an ad banner to be displayed within the Web document.

59. A method of monitoring according to claim 35; wherein the resource located on the server comprises a Web document and includes an embedded URL to another resource located on another server, the other resource comprising an ad banner to be displayed within the Web document.

60. A method of monitoring according to claim 35; wherein the step of requesting the resource located on the server comprises the step of obtaining the resource using a URL.

61. A method of monitoring according to claim 35; wherein the file downloaded from the server is an HTML document.

62. A method of monitoring according to claim 35; wherein the resource located on the server is an HTML document, and the step of requesting the resource comprises the steps of using a TCP/IP connection to pass a request having an HTTP request header containing client identifying indicia to the server.

63. A method according to claim 35; wherein the first executable program is an applet written in JAVA.

64. A method according to claim 35; wherein the software timer of the first executable program commences operation after a predetermined user operation.

65. A method according to claim 35; wherein the first executable program is downloaded in response to a predetermined user operation.

66. A method according to claim 35; further comprising the step of assembling a file in accordance with information stored in the first database.

67. A method according to claim 35; further comprising the steps of acquiring client identifying indicia from the client and storing the client identifying indicia in a second database.

68. A method according to claim 35; further comprising the steps of determining the amount of data downloaded to the client, and storing the amount of data downloaded to the client in the first database.

69. In a computer network having one or more servers connectable to one or more clients, a method for monitoring the interaction of a user with a file downloaded from a server, comprising the steps of:

using a TCP/IP connection to pass a request for an HTML document from a client to a first server using an HTTP protocol, the HTML document containing text and embedded URLs, one or more of the URLs for pointing to a graphical image located on a second server, the image being embedded inside the HTML document using an HTML tag to specify the source URL for an image and predetermined layout parameters, a second URL for pointing to a first executable program that runs on a server, the first executable program being embedded inside the HTML document using an HTML tag to specify the source URL for the program, and being executable upon the server in response to a TCP/IP request by a client, and a third URL for pointing to a second executable program that runs on the client, the second executable program being embedded inside the HTML document using an HTML <APPLET> tag to specify the source URL for the program and being executable on a client in response to a TCP/IP request; downloading the HTML document to the client;

using a TCP/IP connection to fetch the graphical images located on the second server specified by the one or more first URLs embedded in the HTML document;

displaying the text and graphical images on the client in accordance with the formatting and layout parameters specified in the HTML document;

using a TCP/IP connection to fetch the first executable program to execute the first executable program on the server and a return of information output from the first executable program to the client in the form of a transparent GIF image type and obtaining information from the HTTP request header including browser type, at least one of network ID, client ID, time of execution and URL of the HTML document and storing said information in a database indexed by at least one of the network ID of the client machine, the client ID of the client machine, and the URL of the HTML document;

using a TCP/IP connection to fetch the second executable program for execution on the client, wherein the second executable program includes a software timer for determining the amount of time the client spends interacting with the HTML document; and

uploading the time determined by the tracking program to the server.

70. A method according to claim 69; wherein the information output from the first executable program to the client is a transparent image.

25

71. In a computer network having one or more servers connectable to one or more clients, a method of monitoring user interaction with a file downloaded from a server, comprising the steps of:

using a client to specify an address of a resource located on a first server;

downloading a file corresponding to the resource from the first server in response to specification of the address;

26

using the client to monitor the amount of time the user spends interacting with and displaying the file downloaded from the first server;

using a server to acquire client identifying indicia from the client; and

uploading the amount of time determined by the first executable program to a server.

* * * * *

APPENDIX B-3



(19) **United States**

(12) **Patent Application Publication**

Choi et al.

(10) **Pub. No.: US 2003/0236905 A1**

(43) **Pub. Date: Dec. 25, 2003**

(54) **SYSTEM AND METHOD FOR AUTOMATICALLY RECOVERING FROM FAILED NETWORK CONNECTIONS IN STREAMING MEDIA SCENARIOS**

(21) Appl. No.: **10/179,583**

(22) Filed: **Jun. 25, 2002**

Publication Classification

(75) Inventors: **Yejin Choi**, Bellevue, WA (US); **Alexandre Grigorovitch**, Redmond, WA (US); **Troy Batterberry**, Kirkland, WA (US)

(51) **Int. Cl.⁷ G06F 15/16**

(52) **U.S. Cl. 709/231**

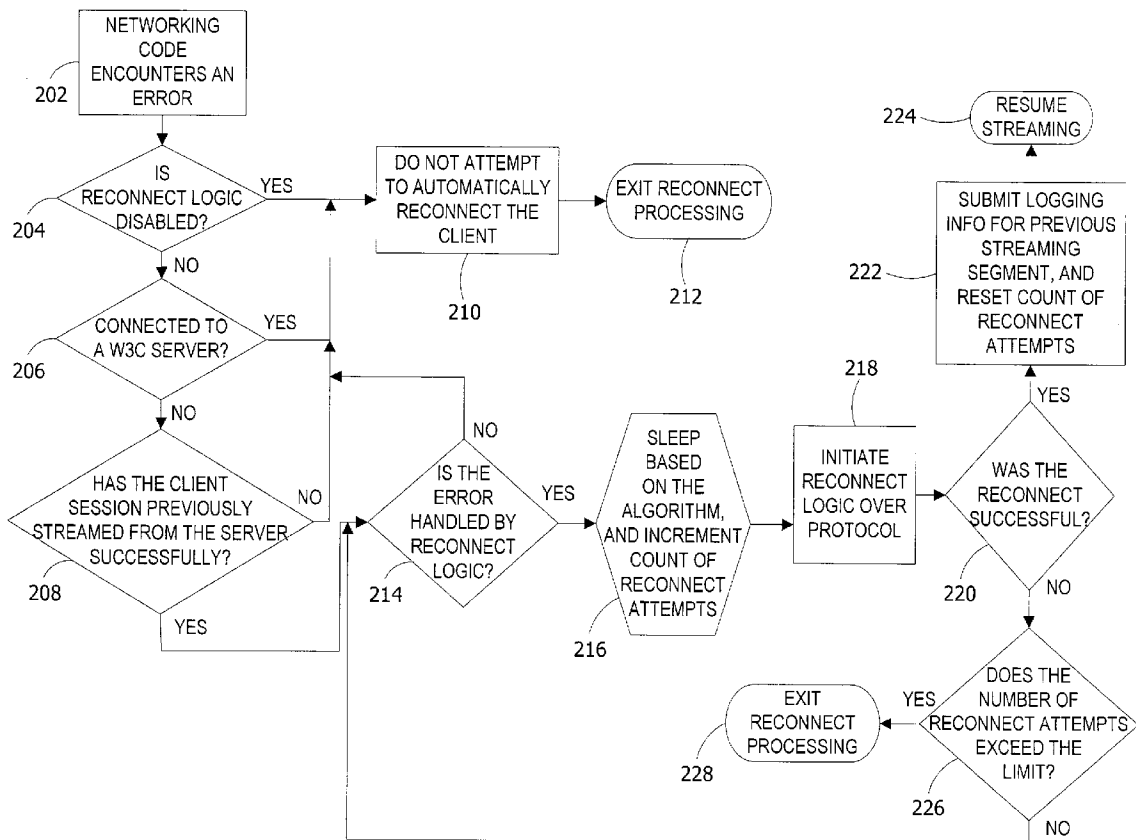
(57) **ABSTRACT**

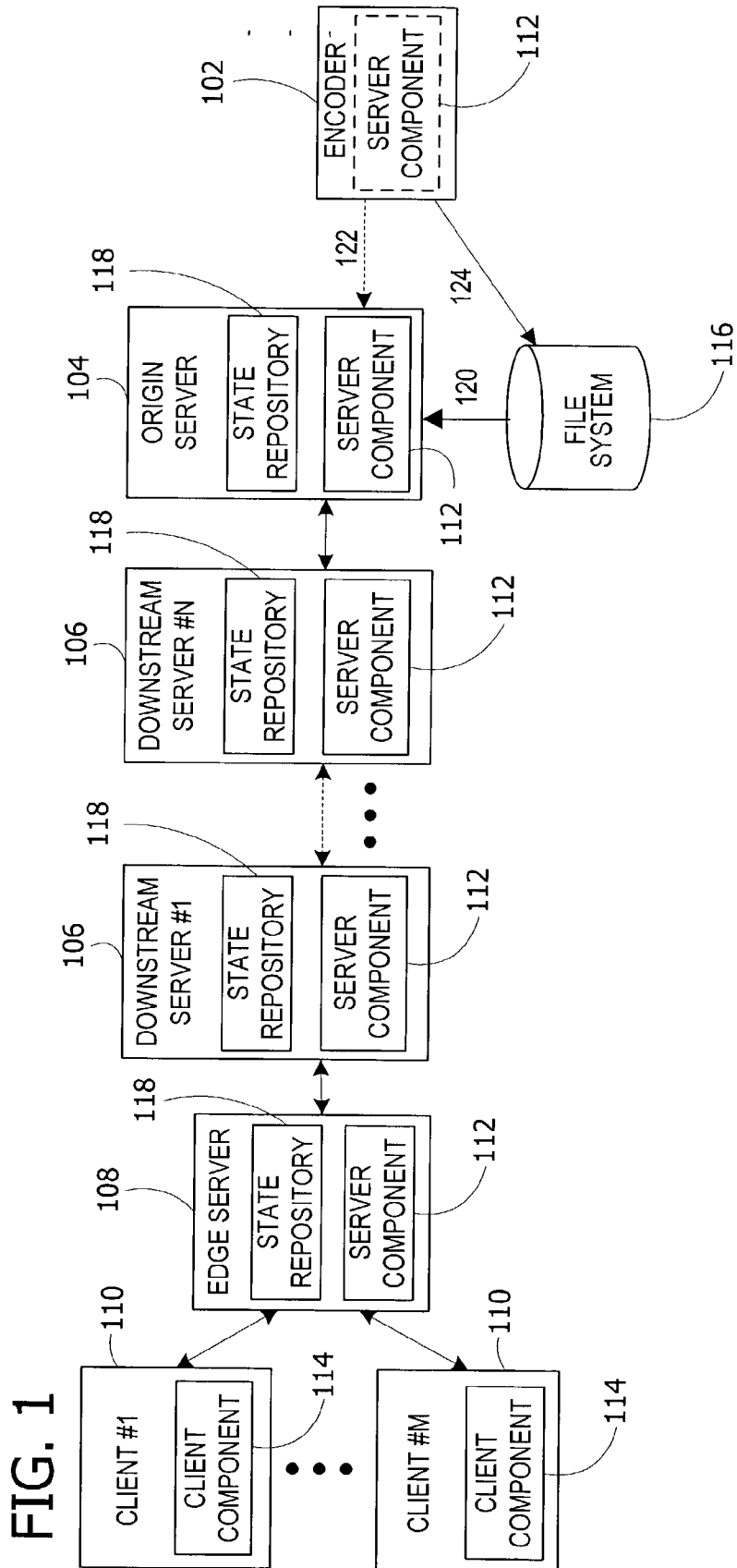
A system and method for automatically recover from broken network connections in streaming media scenarios. Server software executing on the server communicates with client software executing on the client during the streaming media session. If the streaming media session is interrupted, the server software and the client software exchange messages to associate the client with a client state stored by the server and to re-synchronize playback of the content.

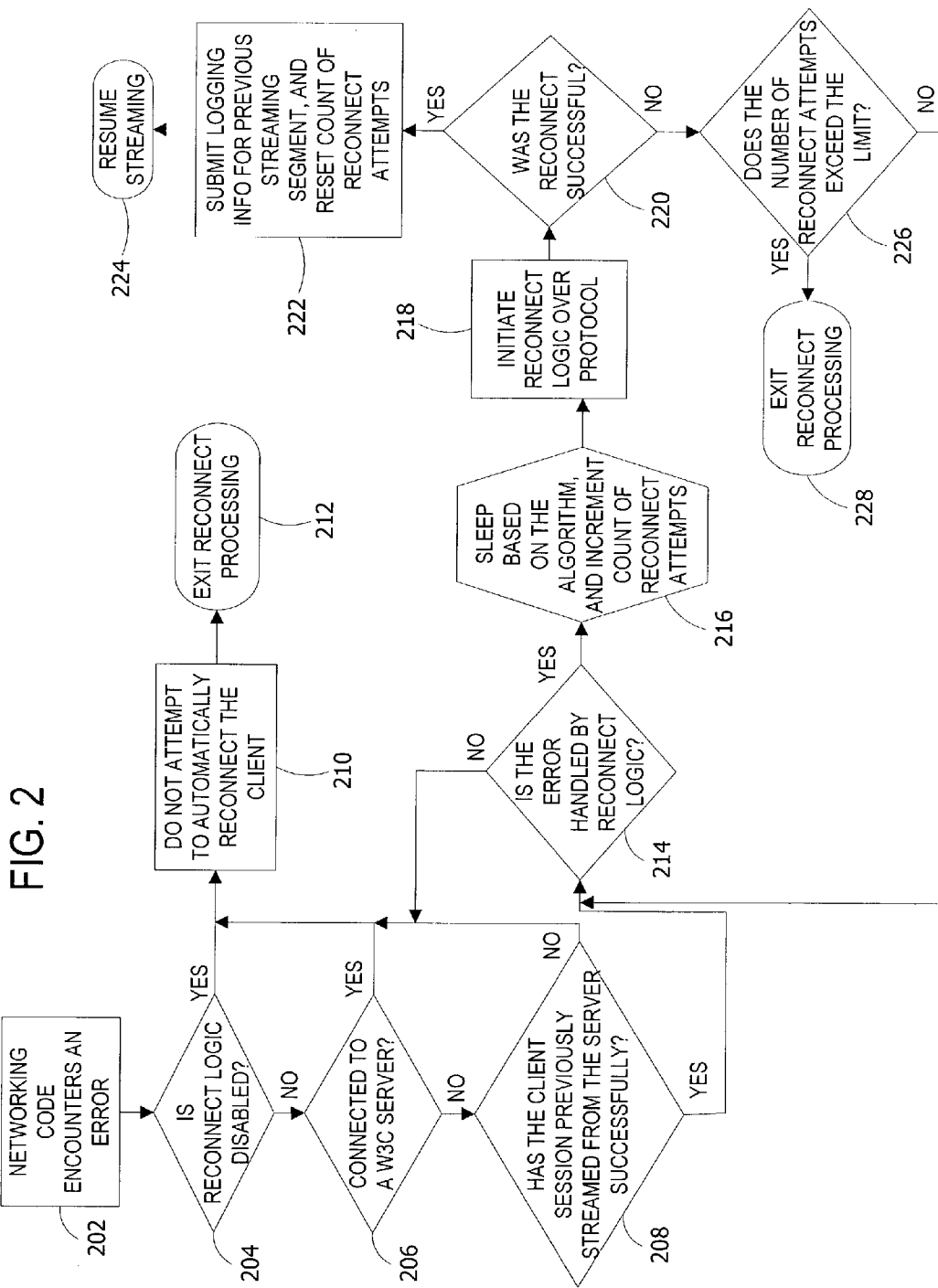
Correspondence Address:

SENNIGER POWERS LEAVITT AND ROEDEL
ONE METROPOLITAN SQUARE
16TH FLOOR
ST LOUIS, MO 63102 (US)

(73) Assignee: **Microsoft Corporation**







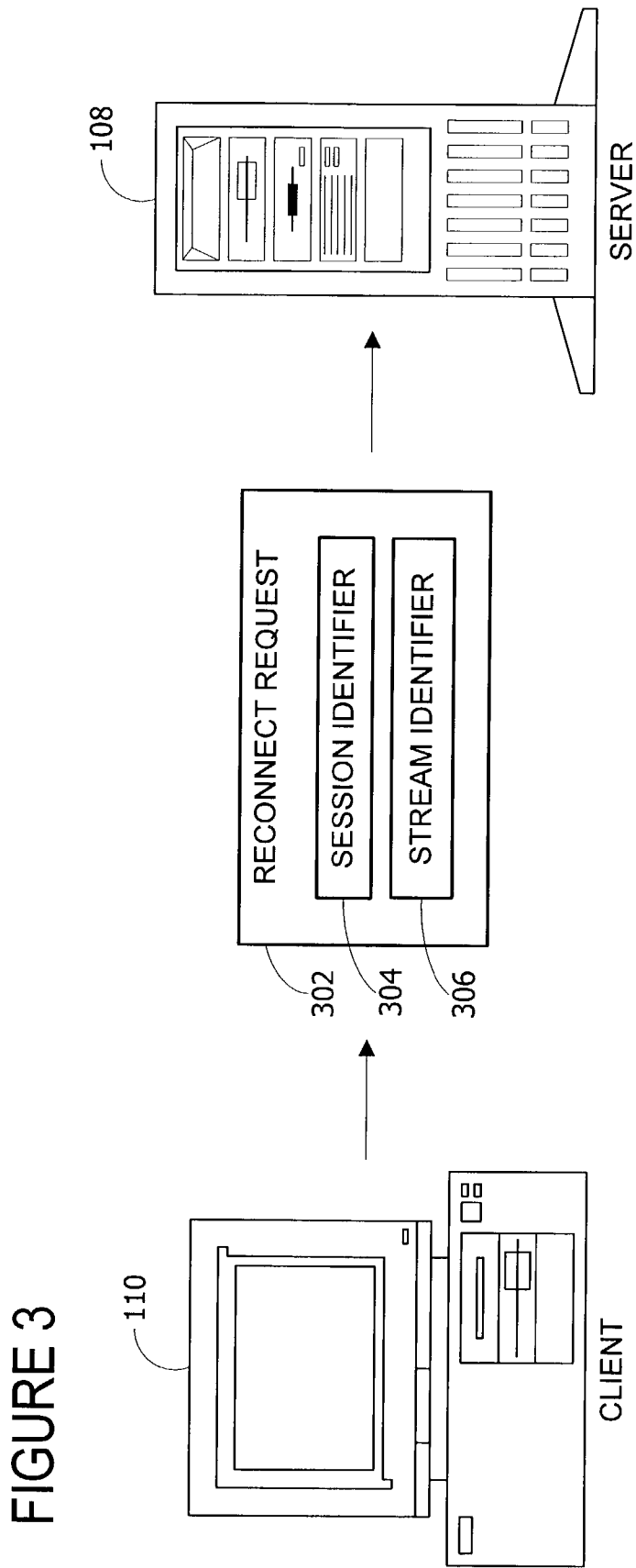


FIGURE 3

FIG. 4

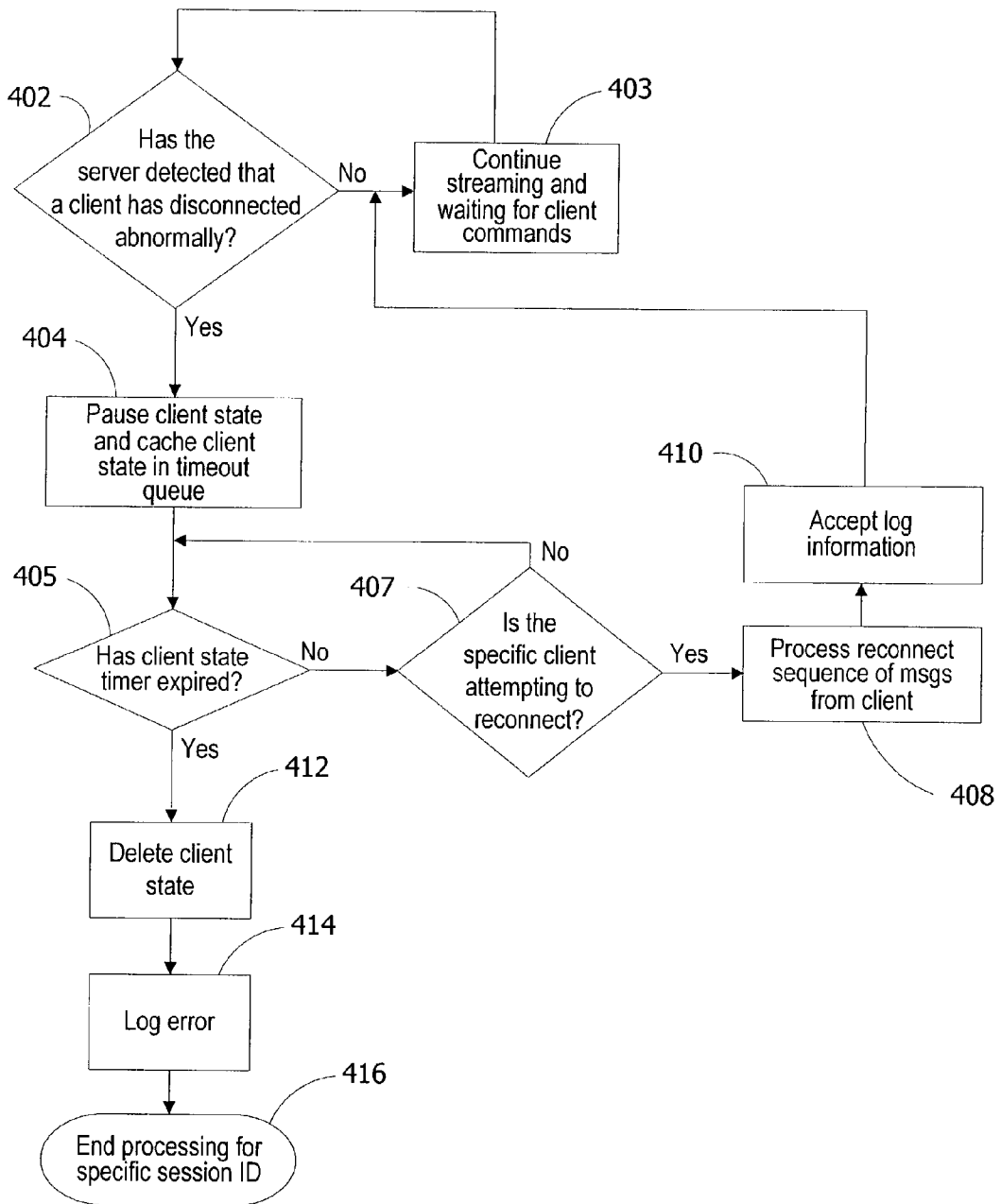


FIG. 5

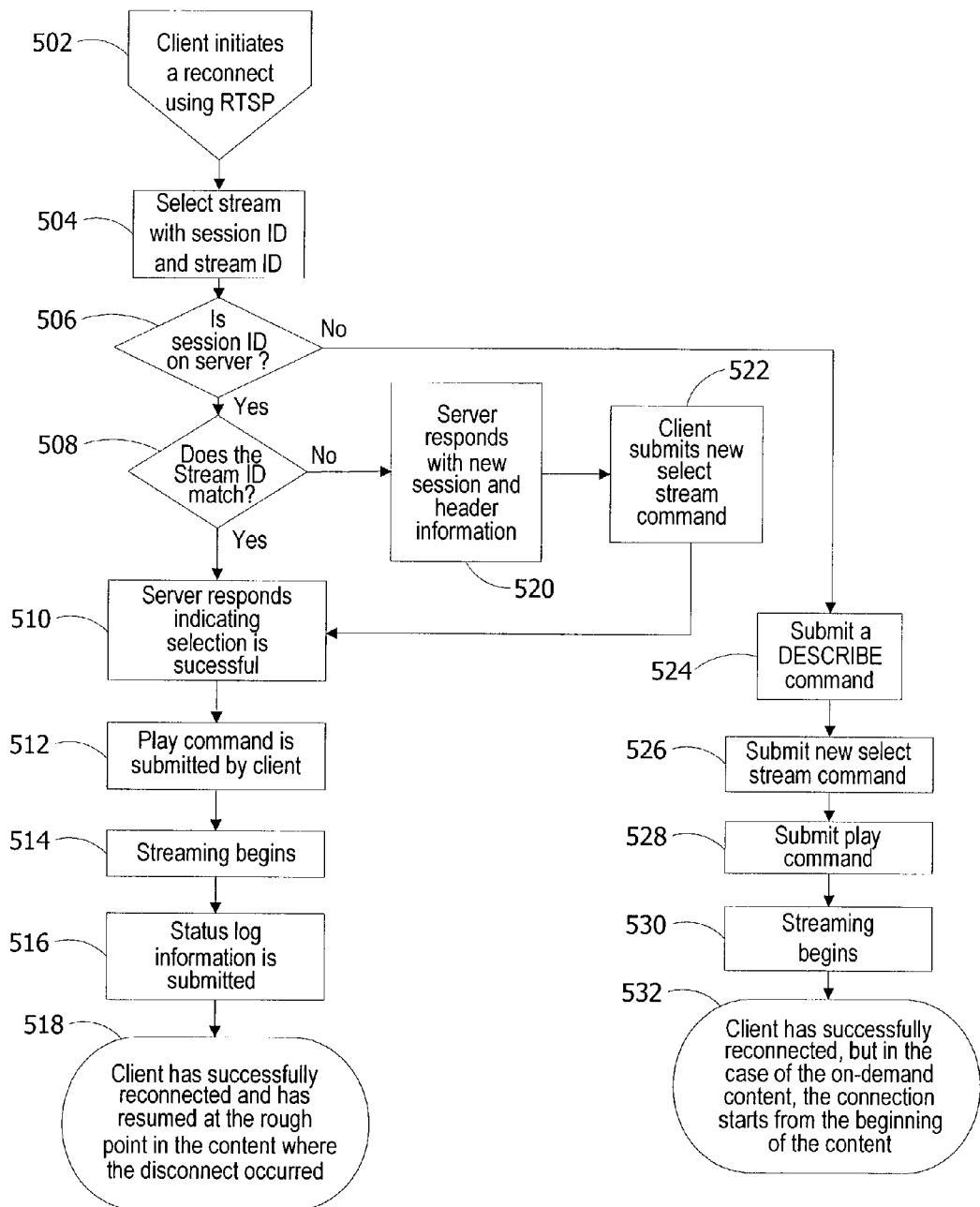
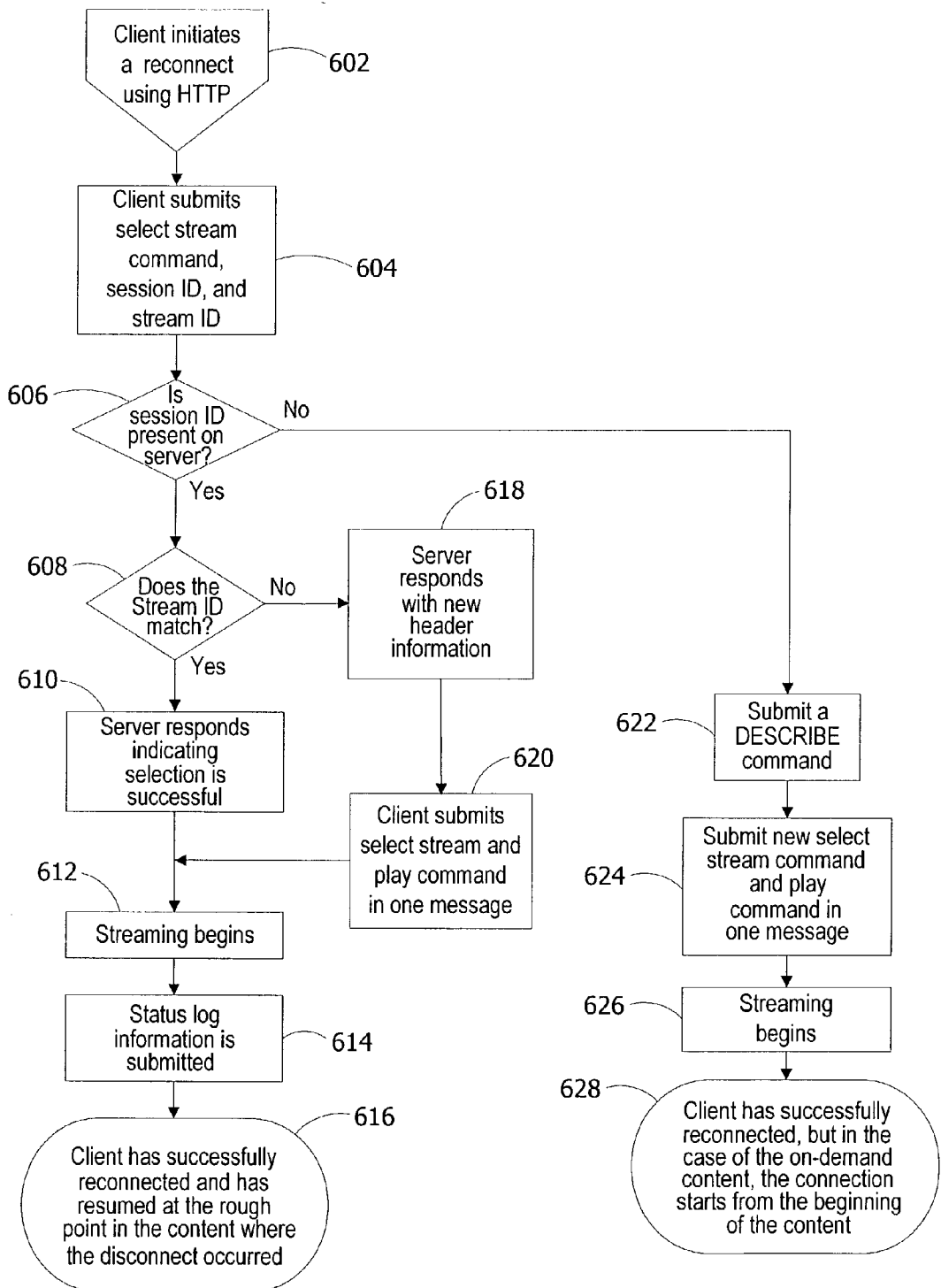
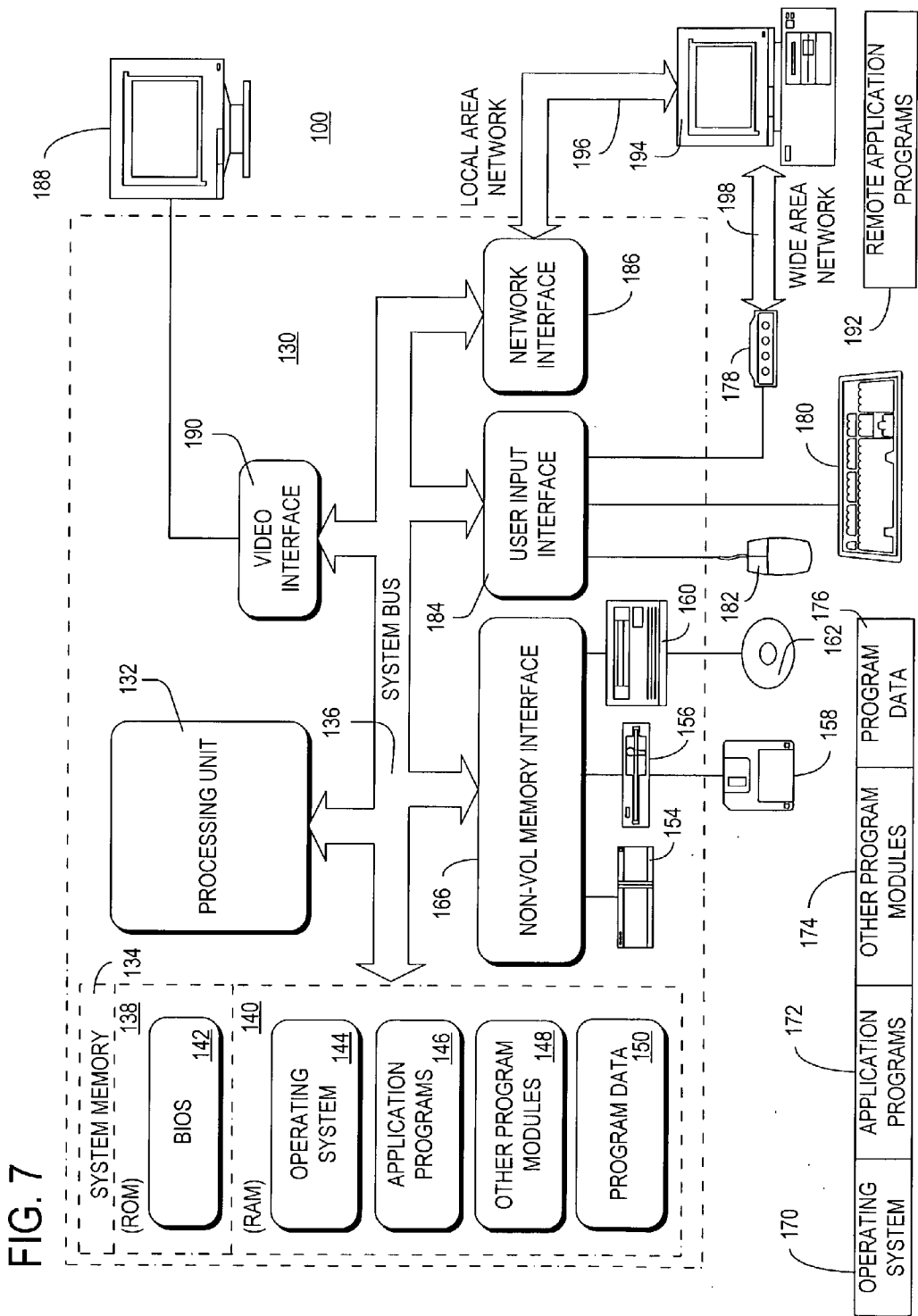


FIG. 6





**SYSTEM AND METHOD FOR AUTOMATICALLY
RECOVERING FROM FAILED NETWORK
CONNECTIONS IN STREAMING MEDIA
SCENARIOS**

TECHNICAL FIELD

[0001] The present invention relates to the field of streaming media. In particular, this invention relates to a system and method for automatically recovering from failed network connections in streaming media scenarios.

BACKGROUND OF THE INVENTION

[0002] Content streaming includes the streaming of audio, video, and/or text data from a network server to a client computer on an as-needed basis. The client computer renders the data as it is received from the network server. For example, audio, video, or audio/visual coverage of noteworthy events can be broadcast with streaming multimedia over a network such as the Internet as the events unfold. Similarly, television and radio stations can transmit live content over the network as streaming multimedia.

[0003] Streaming media over diverse networks poses a variety of technical challenges. The network connection between the server and the client is often subject to adverse conditions such as congestion, packet loss, varying latencies, IGMP/ICMP errors, rebooting routers or other networking devices, rebooting servers, inadvertent reset of TCP connections, lost modem connections, and temporarily unplugged network cables. Depending on the severity of the issue, some streaming media players encounter such adverse conditions and subsequently post a critical error to the user interface. The error is critical in that the user must manually intervene and re-establish the streaming session. Unfortunately, in the case of on-demand content, this also means the user must manually seek to the position in the content that was last being viewed, if seeking in the content is allowed, after the connection is re-established. Further, when this streaming link is disconnected, all the clients and servers that are downstream from the disrupted connection are terminated. The abnormal termination of all downstream clients can result in significant lost revenue.

[0004] For these reasons, a system for automatically recovering from a failed streaming media session is desired to address one or more of these and other disadvantages.

SUMMARY OF THE INVENTION

[0005] The invention includes a method of streaming media content from a server to at least one client. In particular, the invention includes server software executing on the server communicating with client software executing on the client. If the streaming is interrupted, the server software and the client software exchange messages to re-map a state of the client and re-synchronize playback of the content.

[0006] The invention addresses network problems experienced between the client(s) and the server. In addition, the invention addresses network problems experienced by server-to-server and encoder-to-server distribution scenarios, where the server is actually a client streaming from another source. The software of the invention allows a streaming media client player to automatically attempt to

recover from a variety of connection problems with a server without user intervention. Furthermore, the invention software allows the client playing on-demand media to continue after re-connection at roughly the same point in the media program when the connection was lost. The client networking code uses the software of the invention to act upon unexpected errors that are not the direct action of an administrator. The invention includes software on both the server and the client as well as software for a protocol-specific implementation using real-time streaming protocol (RTSP) and hypertext transfer protocol (HTTP).

[0007] With the invention, servers can withstand longer network outages without terminating clients. The invention improves the end-user experience by preventing the user from having to manually recover from connectivity problems. The fault tolerant functionality improves the end user experience for streaming media by more closely mimicking other content delivery metaphors such as television, radio, video cassette recorders, digital versatile disk players, etc.

[0008] In accordance with one aspect of the invention, a method streams media content from a server to at least one client. The method includes establishing a streaming media connection between the server and the at least one client and streaming the media content from the server to the client. The method further includes receiving, by the client, the streamed media content from the server. The method includes sending a reconnect request from the client to the server if the streaming is interrupted. The method also includes receiving, by the server, the reconnect request from the client and re-establishing the streaming media connection with the client. The method includes continuing with the server streaming the media content and the client receiving the streamed media content.

[0009] In accordance with another aspect of the invention, a method stream media content to at least one client. The method includes establishing a streaming media connection with at least one client and streaming the media content to the client. The method also includes receiving a reconnect request from the client if the streaming is interrupted. The method further includes re-establishing the streaming media connection with the client and continuing to stream the media content.

[0010] In accordance with yet another aspect of the invention, a method receives media content streamed from a server. The method includes establishing a streaming media connection with the server and receiving the media content streamed from the server. The method also includes transmitting a reconnect request to the server if the receiving is interrupted. The method further includes re-establishing the streaming media connection with the server and continuing to receive the streamed media content.

[0011] In accordance with yet another aspect of the invention, one or more computer-readable media having computer-executable components in a system wherein a server streams media content to at least one client. The components include a server component and at least one client component. The server component and the client component include computer-executable instructions for exchanging one or more messages to re-map the state of the client and to re-synchronize playback of the content if the streaming is interrupted.

[0012] In accordance with yet another aspect of the invention, one or more computer-readable media store a data

structure representing a reconnect request transmitted by a client to a server to re-establish an interrupted streaming media session. The data structure includes a session identifier identifying the interrupted streaming media session and a stream identifier identifying a media stream streamed by the server to the client in the interrupted streaming media session.

[0013] Alternatively, the invention may comprise various other methods and apparatuses.

[0014] Other features will be in part apparent and in part pointed out hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 is an exemplary block diagram illustrating a streaming media scenario.

[0016] FIG. 2 is an exemplary flow chart illustrating operation of client component autoreconnect software of the invention.

[0017] FIG. 3 is an exemplary block diagram illustrating a client sending a reconnect request to a server.

[0018] FIG. 4 is an exemplary flow chart illustrating operation of server component autoreconnect software of the invention.

[0019] FIG. 5 is an exemplary flow chart illustrating the interaction between the client and the server during reconnection via a real-time streaming protocol.

[0020] FIG. 6 is an exemplary flow chart illustrating the interaction between the client and the server during reconnection via a hypertext transfer protocol.

[0021] FIG. 7 is a block diagram illustrating one example of a suitable computing system environment in which the invention may be implemented.

[0022] Corresponding reference characters indicate corresponding parts throughout the drawings.

DETAILED DESCRIPTION OF THE INVENTION

[0023] Software of the invention provides a mechanism for automatically re-connecting a streaming server with a client if streaming is interrupted during a streaming media session as illustrated in FIG. 1. This invention includes software executing on both the client and one or more servers. In particular, the invention includes server software executing on the server communicating with client software executing on the client. If the streaming is interrupted, the server software and the client software exchange messages to re-map a state of the client and re-synchronize playback of the content.

[0024] Referring to FIG. 1, an exemplary block diagram illustrates a streaming media scenario. The invention software is operable in a system having an optional encoder 102, an origin server 104, one or more downstream servers 106 such as downstream server #1 through downstream server #N, an edge server 108, and one or more clients 110 such as client #1 through client #M. The origin server 104, the downstream servers 106, and the edge server 108 each execute a server software component 112 while the clients 110 execute a client software component 114. The server

component 112 and the client component 114 include computer-executable instructions for exchanging one or more messages to re-map the state of the client 110 and to re-synchronize playback of the content if the streaming is interrupted. Separate state repositories 118 such as a state repository stored by the origin server 104, a state repository stored by the downstream servers 106, and a state repository stored by the edge server 108 store a state of the downstream server 106 or client 110. For example, the edge server 108 stores a state of the client 110. In addition, each of the downstream servers 106 and the origin server 104 store a state for downstream servers acting as clients. For example, the downstream server #1 stores a state of the edge server 108. Similarly, the origin server 104 stores a state of the downstream server #N.

[0025] The origin server 104 is the first server the content flows through on the way to the client 110. The origin server 104 generally receives content from either a file system 116 at 120 or a feed from the encoder 102 at 122. The encoder 102 stores the encoded content in the file system 116 at 124. If the origin server 104 receives content from the encoder 102, the file system 116 may be bypassed, or the encoded content may be concurrently stored in the file system 116 at 124. The downstream servers 106 generally receive data from the origin server 104. In complex distribution scenarios involving multiple levels of servers, the downstream servers 106 may receive and forward content from another server that is sourcing content from the origin server 104. Since the data flows from the origin server 104 to the client 110, a server is considered downstream from previous servers. The edge server 108 is generally the last server in a distribution scenario. The edge server 108 is downstream from all other servers in the distribution chain. The edge server 108 is intended to have direct client connections. For clarity and simplicity, the edge server 108 will be referred to herein as server 108, noting that the invention is operable with any configuration and/or number of servers 104, 106, 108.

[0026] In addition, the edge server 108 maintains a state repository storing a client viewer state of each of the clients 110 (e.g., storing logging statistics). The clients 110 transmit their states to the edge server 108 for storage. The state of each client 110 is maintained for a preset time period after a network failure or other interruption in the streaming.

[0027] In one embodiment, the origin server 104 streams the media content from the file system 116. In an alternative embodiment, the encoder 102 also executes the server component 112 to stream content to the origin server 104 as it is encoded. In such an embodiment, the file system 116 may be bypassed, or the encoded content may be concurrently stored in the file system 116. Those skilled in the art will appreciate that the invention is not limited to the exemplary block diagram of FIG. 1. Instead, it is contemplated by the inventors that the software of the invention is operable in various other client-server streaming media scenarios not specifically described herein.

[0028] The clients 110 may render or otherwise display or process the received content via a media player user interface (UI). Clients 110 receiving streamed media content for long periods of time often encounter a variety of network problems that result in the server-to-client connection or session being lost. With other systems, a lost connection requires user intervention to re-establish the link. With the

software of the invention, the clients **110** and the servers **108** attempt to automatically reconnect. If the server **108** was streaming stored content (e.g., from a computer-readable medium) prior to the session failure, the client **110** can resume playback at the location in the stream when the failure occurred using statistics saved prior to the failure. If the server **108** was streaming live content (e.g., directly from the encoder **102**) prior to the session failure, the client player UI may not receive and render the content that was streamed during the reconnection process. If the reconnection process occurred relatively quickly, the server **108** may have buffered a small amount of the live content, and will deliver that buffered content to the client **110** if reconnection is successful. As such, a user may experience minimal disruption in the playback.

[**0029**] In one embodiment, communication between the servers **108** and client **110** in **FIG. 1** is implemented using a real-time streaming protocol (RTSP) and a session description protocol (SDP). RTSP, as described in the Internet Engineering Task Force (IETF) RFC 2326, the entire disclosure of which is incorporated herein by reference, is an application-level protocol for control of the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as a user datagram protocol (UDP), a multicast UDP and a transmission control protocol, and provide a means for choosing delivery mechanisms based upon a real-time transport protocol.

[**0030**] For example, the Real-time Transport Protocol (RTP), as described in the IETF RFC 1889, the entire disclosure of which is incorporated herein by reference, provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers.

[**0031**] SDP, as described in the IETF RFC 2327, the entire disclosure of which is incorporated herein by reference, is an application level protocol intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. SDP can be used in conjunction with RTSP to describe and negotiate properties of the multimedia session used for delivery of real-time data.

[**0032**] The invention software supports automatic reconnection logic **112**, **114** for various protocols such as HTTP (see **FIG. 6**), RTSP (see **FIG. 5**), and any proprietary protocols in the client component **114** and the server component **112**. The invention software also logs the first segment of information received following a successful reconnect (e.g., as a status code of **210**). The invention software supports broadcast and on-demand modes of opera-

tion. The automatic reconnection logic **112**, **114** can be tuned/disabled in the server **108** (e.g., to act as a distribution client) and in the client **110**. The invention software staggers the client reconnect attempt requests over time to prevent the server **108** from being overwhelmed by thousands of simultaneous reconnect requests. The reconnecting client **110** is authenticated and authorized if corresponding security is enabled. The reconnecting client **110** resumes at the same point of a seekable on-demand playlist element. In one embodiment, the server **108** maintains a client viewer state if data has actually been streamed. This check increases the difficulty of developing a denial of service attack. A disconnection resulting from a client inactivity timeout on the server **108** does not result in an error immediately displayed on the client **110**. Instead, the client **110** attempts to re-open the file at the beginning of the playlist once play is pressed. In one embodiment, a seek is not possible because the client viewer state on the server **108** for the previous connection will no longer be present. In embodiments lacking a client viewer state present on the server **108**, seeking to the previous playlist entry element in a server-side playlist may be disabled. An error displays on the client **110** if the re-open attempt is unsuccessful.

[**0033**] In one embodiment, the invention software does not attempt to automatically reconnect when an administrator for the server **108** terminates a connection, when the server **108** denies access due to an authentication failure, when playing content from a web server, or when the server **108** denies access due to an authorization failure.

[**0034**] In operation, client **110** and server **108** computers such as computer **130** execute computer-executable instructions such as those illustrated in **FIG. 2** and **FIGS. 4-6** to re-establish a streaming media connection between the server **108** and the client **110**. The server **108** streams the media content to the client **110**. The client **110** receives the streamed media content from the server **108**. If the streaming is interrupted, the client **110** sends a reconnect request to the server **108**. The server **108** receives the reconnect request from the client **110**. The server **108** and the client **110** re-establish the streaming media connection. Re-establishing includes the server **108** mapping a reconnecting client **110** with a state maintained by the server **108**. Alternatively, re-establishing includes creating a new session for streaming if no maintained state corresponds to the client **110**. The server **108** continues streaming the media content to the client **110** over the re-established streaming media connection.

[**0035**] Client Component Software

[**0036**] Referring next to **FIG. 2**, an exemplary flow chart illustrates operation of client component autoreconnect software **114** the invention. The client component software **114** acts upon unexpected errors at **202** that are not the direct action of an administrator. The client component software **114** operates if the client **110** has successfully streamed from the server **108** previously at **208** and the error is handled by reconnect logic **114** at **214**.

[**0037**] If thousands of clients **110** attempt to auto-reconnect at exactly the same time, the server **108** may not be able to process any of them successfully. Also, repeated reconnect attempts can tax the client's processor. Therefore, the software of the invention spreads out the timing of the auto-reconnect requests by clients **110**. To prevent all clients

110 from overwhelming a streaming media server **108** with a flood of reconnect requests at exactly the same time, the client **110** employs software to sleep at **216** between reconnect attempts. The sleep duration involves a random component to help spread reconnect requests when multiple clients **110** are disconnected at the same time. The sleep software is also used to minimize the amount of client processing required to successfully reconnect. For example, if a client **110** continuously reconnects while waiting for a router to reboot, it could adversely affect the client processor load. By delaying the transmission of the reconnect request to the server **108** for a preset time period between reconnect attempts, both the client **110** and the server **108** are optimized. For example, the client software may wait for five seconds between failed reconnect attempts and increment a reconnect counter for each attempt. In one embodiment, the client **110** attempts to reconnect twenty-five times before halting. That is, if the reconnect counter exceeds a preset threshold at **226**, the client software halts the reconnect attempt and logs an error at **228**.

[**0038**] The number of attempts the client **110** retries to connect is fully configurable through a client application programming interface (API) and also a uniform resource locator (URL) modifier. A URL modifier allows a content provider or other encoder such as encoder **102** to control the number of reconnect attempts made by the client **110** so that it is appropriate for the environment. An example of the URL modifier follows.

[**0039**] `mms://server/file.asf?WMReconnect=15`

[**0040**] In this example, the client **110** will attempt to reconnect fifteen times (e.g., at **218**) before failing with an error. If the client software successfully reconnects with the server **108** at **220**, logging statistics are sent to the server **108**, the reconnect counter is reset to zero at **222**, and streaming resumes at **224**.

[**0041**] There are several mechanisms that trigger the client **110** to attempt a reconnect. A network error detected from the local protocol stack or the error signal sent by the server **108** or prolonged no data period (e.g., a starvation timeout) will potentially trigger the reconnect logic **114**. If the error signal sent by the server **108** denotes that the server **108** intended to disconnect the client **110** deliberately, the client **110** will not attempt to reconnect. The client **110** will attempt to reconnect even in a paused state in order to maintain the client viewer status active at the server **108**. The player code fires events to update the status of the player user interface to indicate when the client **110** has started (and finished) reconnecting.

[**0042**] The client **110** does not attempt to automatically reconnect with the server **108** under various conditions such as when the client component **114** and/or the server component **112** is disabled at **204**. In one embodiment, the client **110** does not attempt to automatically reconnect with the server **108** when the server **108** is a World Wide Web Consortium server at **206**. Under such conditions, the client **110** and the server **108** do not automatically reconnect at **210** and reconnect processing exits at **212**.

[**0043**] In a server distribution or a cache/proxy scenario where one server is receiving content from the origin server **104**, the downstream server **106** is essentially a client such as client **110** in that it is streaming content from the origin

server **104**. In this scenario, the downstream server **106** can employ auto-reconnect software to connect back to the origin server **104** using software similar to the software **114** used by the client **110**.

[**0044**] Referring next to **FIG. 3**, an exemplary block diagram illustrates the client **110** sending a reconnect request **302** to the server **108** to re-establish an interrupted streaming media session. In the exemplary embodiment of **FIG. 3**, the reconnect request **302** is a data structure including a stream identifier **306** and a session identifier **304**. The session identifier identifies the interrupted streaming media session. For example, the session identifier may be a 64-bit or a 32-bit value generated by the server **108** and identifies the client-server relationship. The stream identifier identifies a media stream streamed by the server **108** to the client **110** in the interrupted streaming media session. For example, the stream identifier may be a 32-bit value generated by the server **108** to identify a particular stream in the media content.

[**0045**] Server Component Software

[**0046**] Referring next to **FIG. 4**, an exemplary flow chart illustrates operation of the server component autoreconnect software **112** of the invention. During the period in which the server **108** does not detect at **402** that the client **110** has disconnected abnormally, the server **108** continues streaming at **403** and waiting for commands from the client **110**. If the server **108** detects at **402** that the client **110** has disconnected abnormally, the server **108** employs a variety of mechanisms to allow the client **110** to reconnect. These mechanisms are described below.

[**0047**] The client **110** periodically transmits state data (e.g., logging statistics) to the server **108** for storage. In addition, the server **108** tracks the status of each client viewer state and allows an administrator of server **108** to determine the state of any client **110**. The state data includes a session identifier and a stream identifier corresponding to the current client-server session and the streams being delivered, respectively. The server **108** pauses the client state and maintains the client viewer state for a pre-determined (e.g. configurable) duration or time period at **404**. The client viewer state may be stored or cached in the state repository, a timeout queue, or the like. Since the client viewer state consumes server resources, the server **108** will not maintain the state indefinitely. After determining that the configurable duration expired at **405**, the server **108** removes the client viewer state at **412**, frees the associated resources, logs an error at **414**, and ends processing at **416** for the current session. For example, logging an error at **414** includes the server **108** generating a log on behalf of the client **110** because the reconnecting client **110** will not submit a log (e.g., with status code **210**) for content rendered before the reconnect event.

[**0048**] If the client **110** attempts to re-connect or otherwise re-establish a connection while the client viewer state is present on the server **108** at **405**, the client **110** end-user experience is optimal. If the server **108** determines at **407** that the client **110** attempting to reconnect is associated with a cached client state, the server **108** processes at **408** the reconnect sequence of messages from the client **110**.

[**0049**] The server **108** accepts logging information at **410** from the previous session from the clients **110** that re-

connect. For example, a client such as client **110** that streams content for one hour loses its connection to the server **108** prior to successfully submitting logging information. Through the invention software, the client **110** reestablishes the connection back to the server **108** and submits the logging information for the previous segment in addition to continuing with the streaming process. Logging information is data that describes the characteristics of the client **110** and the rendering information associated with the streaming session. Logging information includes, but is not limited to, packet loss statistics and frame rate rendered. See Appendix C for an exemplary list and discussion of logging statistics.

[0050] For example, if the client viewer state is available at the server **108** by the time the client **110** recovers the connection, and if the client **110** is reconnecting in streaming status, the client **110** will submit a log with status code **210**. Apart from the status code, the content of the log is the same as a regular log sent after playback. If the preset time period has elapsed, the server component **112** deletes the client viewer state. After accepting the log from the client **110** at **410**, the server **108** resumes streaming at **403**.

[0051] If the disconnection was the specific intention of the server **108** and not due to an unforeseen fault, the server **108** will inform the client **110** before disconnecting so that the client **110** does not try to reconnect unnecessarily. An example of this might be when an administrator for server **108** terminates a broadcast program normally. If the specific client viewer state was for the content which requires authentication, the server **108** will re-challenge the reconnecting client **110**.

[0052] Referring next to FIG. 5, an exemplary flow chart illustrates the interaction between the client **110** and the server **108** during reconnection via a real-time streaming protocol. In the embodiment illustrated in FIG. 5, the software of the invention is implemented with RTSP. If an RTSP client such as client **110** is attempting to reconnect at **502** in streaming status, the RTSP client **110** sends at **504** multiple SETUP messages (e.g., reconnect requests **302** with the session identifier **304** and the stream identifier **306**) for SelectStreams to re-configure the data ports and stream parameters. If there is an RTSP proxy, some of the parameters may get reset. Attempting to re-establish the session includes the server **108** searching for the received session identifier in the state repository. If the received session identifier is found at **506** within the state repository, the server **108** searches at **508** for the received stream identifier within the state repository. If SelectStreams succeeds (e.g., the session identifier and stream identifier are found within the state repository), the server responds at **510** indicating that the selection was successful. In addition, the client **110** sends a PLAY command at **512** to restart streaming at **514**. If the original viewer state could be retrieved, the client **110** sends a log message at **516** (e.g., with a status of **210**) to report the play status before reconnect after the PLAY command completes. The streaming resumes at **518** at the approximate point in the content where the disconnect or other error occurred.

[0053] If the received stream identifier is not found within the state repository, the server **108** transmits at **520** one or more other stream identifiers to the client **110** for selection by the client **110**. The other stream identifiers include the stream identifiers for any content available from the server

108, including the streams that may have been streaming during the failed session. The client **110** transmits at **522** a playback request to the server **108** where the playback request specifies at least one of the other stream identifiers. The server **108** then streams the media content associated with the stream identifiers selected by the client **110**.

[0054] If the server **108** does not have the viewer state for the requested session at **506** (e.g., the session identifier is not in the state repository), the server **108** responds with an error to indicate the session was lost. In this case, the client **110** attempts to re-establish the connection by submitting a DESCRIBE command at **524** to retrieve the most recent streaming description and then submits a SelectStream command at **526** and a Play command at **528** based on the new description. If the viewer status is available at the server **108** but the streaming description that the client **110** retrieved before being disconnected is no longer current, the server **108** pushes the most recent information of the requested URL by submitting Announce right after accepting Play. If an RTSP client **110** is reconnecting in paused status, it sends SelectStreams to re-configure data ports and stream parameters. The client **110** sends periodic GET_PARAMETERS for KeepAlives to keep the viewer state active until the user wants to play again. The command SelectStream may fail if the requested session on the server **108** was gone, in which case client **110** will submit DESCRIBE and retrieve the most recent streaming description. In this specific example, there will be no **210** log message report after reconnect. When streaming begins at **530**, the client **110** has successfully reconnected. In the case of on-demand content, the streaming starts from the beginning of the content.

[0055] Referring next to FIG. 6, an exemplary flow chart illustrates the interaction between the client **110** and the server **108** during reconnection via a hypertext transfer protocol. The flow in FIG. 6 is generally similar to that described in FIG. 5. In the embodiment illustrated in FIG. 6, the software of the invention is implemented in HTTP. If an HTTP client such as client **110** is attempting to reconnect at **602** in streaming status, the client **110** sends one GET command at **604** that contains both SelectStreams and Play information along with the session identifier **304** and the stream identifier **306**. The server **108** attempts to associate the maintained client viewer state with the client **110** sending the reconnect request **302**. If the server **108** determines that the original viewer state on the server **108** still exists (i.e., the session identifier **304** is present on the server **108** at **606** and the stream identifier **306** is present on the server **108** at **608**), the server **108** responds to the client **110** indicating that the selection was successful at **610**. Streaming begins at **612**. The client **110** sends a log message at **614** (e.g., with a **210** status code) to report the play status before the reconnect event. Whether the requested viewer state is available or not, the server **108** does not return an error as in the RTSP implementation. If the requested viewer state is not available, the server **108** handles the request **302** based on the most recent streaming description of the requested URL. The server **108** includes the most recent streaming description and the viewer state information in the response so that the client **110** can detect the current status of the server **108**. That is, the server **108** responds with new header information at **618**. The client **110** submits a select stream and play command in one message at **620** and streaming begins at **612**.

[0056] If the HTTP client **110** is reconnecting in a paused status, the client **110** sends OPTIONS for KeepAlives to keep the viewer state active until the user wants to play again. In this exemplary implementation, there are no log messages (e.g., with a status code of **210**) reported after reconnect.

[0057] If the client viewer state is in the state repository accessible by the server **108**, the client **110** attempts to automatically reconnect to the same session when the connection is reestablished, as shown in the network trace listed in Appendix A.

[0058] When a client **110** attempts to automatically reconnect to the same session after a network outage, the session may have expired at **606**. In this case, the client **110** makes a new attempt to connect, this time without including the session identifier. That is, the client **110** submits a DESCRIBE command at **622**. The server **108** creates a new session and returns the identifier, as shown in the network trace listed in Appendix B. The client **110** submits a new select stream command and play command in one message at **624** and streaming begins at **626**. The client **110** has successfully reconnected at **628**. In the case of on-demand content, the streaming starts from the beginning of the content.

[0059] Errors Handled by Auto-Reconnect Software

[0060] Errors handled by the auto-reconnect software include, but are not limited to, the following errors. If any of the errors listed below initially occur, the reconnect software will be triggered:

- [0061] ERROR_CONNECTION_ABORTED
- [0062] ERROR_NETNAME_DELETED
- [0063] ERROR_CONNECTION_INVALID
- [0064] NS_E_TIMEOUT
- [0065] NS_E_PROXY_TIMEOUT
- [0066] NS_E_NOCONNECTION
- [0067] NS_E_NET_READ
- [0068] NS_E_CONNECTION_FAILURE
- [0069] WSAECONNRESET
- [0070] WSAECONNABORTED
- [0071] WSAENETUNREACH
- [0072] WSAENETDOWN

[0073] If any of the errors below occur during a reconnect attempt, the reconnect software is repeated (assuming the maximum number of attempts has not been reached):

- [0074] ERROR_OPERATION_ABORTED
- [0075] ERROR_NETWORK_UNREACHABLE
- [0076] ERROR_HOST_UNREACHABLE
- [0077] ERROR_PROTOCOL_UNREACHABLE
- [0078] NS_E_SERVER_DNS_TIMEOUT
- [0079] NS_E_PROXY_DNS_TIMEOUT
- [0080] NS_E_SERVER_NOT_FOUND
- [0081] NS_E_PROXY_NOT_FOUND

[0082] NS_E_CANNOTCONNECT

[0083] NS_E_CANNOT_CONNECT_TO_PROXY

[0084] WSAEHOSTUNREACH

[0085] WSAETIMEDOUT

[0086] The auto-reconnect software **112**, **114** is not invoked for a variety of other errors. The list of errors or conditions that do not result in a reconnect attempt against the server **108** includes, but is not limited to, a publishing point limit is reached, the client **110** fails authentication, the title is not found, the server **108** or publishing point is denying new connections, the publishing point is stopped, the server **108** does not initially respond in time, the administrator for the server **108** terminates the client **110**, the server **108** inactivity timeout feature disconnects the player, the reconnect software is disabled, and the server **108** is a World Wide Web Consortium server.

[0087] Logging During an Auto-Reconnect

[0088] Logging statistics are used by content distribution networks (CDNs) to bill customers. As a result, accurate logging statistics are critically important for the CDNs to maximize their revenue opportunities. See Appendix C for an exemplary list and discussion of logging statistics. A complete log entry (e.g., defined by the status code **200** or **210**) reflects what the client **110** actually rendered. There are several possible cases that may occur during the streaming of media such as described in the following examples. Those skilled in the art will note that the status codes are merely exemplary, and do not limit the logging aspects of the invention in any way.

[0089] The content may be streamed successfully without the loss of the connection between the server **108** and the client **110**. In this case, the auto-reconnect software is not used and a normal log entry is written.

[0090] In another scenario, a server-client connection or a distribution connection may be temporarily lost for a short period of time, but then automatically re-established. In this case, two log entries are written. One log entry contains the information regarding the content received and played by the client **110** prior to the disconnect event. For example, this log entry has a status code of **210**. The client **110** information for this log entry is submitted during the handshake for the reconnect request **302**. Another log entry occurs following the successful completion of the content. This log entry includes information for the duration of the clip streamed immediately after the reconnect occurred. For example, this log entry has a normal status code of **200**.

[0091] In another example, the server-client connection or the distribution connection may be lost and auto-reconnect software **112**, **114** is either disabled or unable to reconnect within the allotted number of attempts. This situation results in one log entry with the status code of **408**. The entry includes information regarding the segment of content played prior to the disruption.

[0092] Distribution Outages and Client Buffering

[0093] In an alternate scenario of the invention, during a distribution outage, the clients **110** do not receive any streamed data. As a result, the starvation timer on the clients **110** may eventually fire and ultimately result in all the clients **110** attempting to reconnect to the server **108**. This situation

is undesirable because it greatly increases the load on the server **108** and lengthens the time required for the clients **110** to recover from the outage. To preclude this situation, software of the invention operating on the server **108** fakes a stream switch that places the clients **110** in a waiting state to prevent starvation during a distribution outage. When the distribution connection recovers, the server software **112** sends another stream header before streaming the content. This mechanism allows the clients **110** to resume playing.

[0094] Configurable Settings

[0095] In one embodiment, the server **108** namespace is used to configure the duration a client state is maintained on the server **108** after an abnormal disconnect. The following exemplary namespace parameters tune these timeout values.

[0096] “ClientIdTimeoutForPlayer”—(60 sec default)

[0097] “ClientIdTimeoutForPull” (60 sec default—distribution connections)

[0098] “ClientIdTimeoutForPush” (300 sec default—encoder connections)

[0099] Additionally, the software exposes a property (e.g., `AutoReconnectLimit`). A value of zero disables the auto-reconnect logic **114**. A value of (-1) results in autoreconnect software attempting to reconnect forever. In addition, the client software **114** fires events such as `WMT_RECONNECT_START` and `WMT_RECONNECT_END`, during the reconnect process. This information is available to the higher level player application for display in the UI.

[0100] Client Options

[0101] The client software exposes an object model property (e.g., `AutoReconnect`). The object model property is adjustable from the default player UI. In one embodiment, the default value for this property is three. A value of zero disables the auto-reconnect software and a value of (-1) results in auto-reconnect software attempting to reconnect forever. In addition, the player UI processes events such as `WMT_RECONNECT_START` and `WMT_RECONNECT_END` during the reconnect process. This information is then displayed in the player UI.

[0102] Exemplary Operating Environment

[0103] **FIG. 7** shows one example of a general purpose computing device in the form of a computer **130**. In one embodiment of the invention, a computer such as the computer **130** is suitable for use in the other figures illustrated and described herein. Computer **130** has one or more processors or processing units **132** and a system memory **134**. In the illustrated embodiment, a system bus **136** couples various system components including the system memory **134** to the processors **132**. The bus **136** represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0104] The computer **130** typically has at least some form of computer readable media. Computer readable media, which include both volatile and nonvolatile media, removable and non-removable media, may be any available medium that can be accessed by computer **130**. By way of example and not limitation, computer readable media comprise computer storage media and communication media. Computer storage media include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. For example, computer storage media include RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to store the desired information and that can be accessed by computer **130**. Communication media typically embody computer readable instructions, data structures, program modules, or other data in a data signal such as a carrier wave or other transport mechanism and include any information delivery media. Those skilled in the art are familiar with the data signal, which has one or more of its characteristics set or changed in such a manner as to encode information in the signal. Wired media, such as a wired network or direct-wired connection, and wireless media, such as acoustic, RF, infrared, and other wireless media, are examples of communication media. Combinations of the any of the above are also included within the scope of computer readable media.

[0105] The system memory **134** includes computer storage media in the form of removable and/or non-removable, volatile and/or nonvolatile memory. In the illustrated embodiment, system memory **134** includes read only memory (ROM) **138** and random access memory (RAM) **140**. A basic input/output system **142** (BIOS), containing the basic routines that help to transfer information between elements within computer **130**, such as during start-up, is typically stored in ROM **138**. RAM **140** typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit **132**. By way of example, and not limitation, **FIG. 7** illustrates operating system **144**, application programs **146**, other program modules **148**, and program data **150**.

[0106] The computer **130** may also include other removable/non-removable, volatile/nonvolatile computer storage media. For example, **FIG. 7** illustrates a hard disk drive **154** that reads from or writes to non-removable, nonvolatile magnetic media. **FIG. 7** also shows a magnetic disk drive **156** that reads from or writes to a removable, nonvolatile magnetic disk **158**, and an optical disk drive **160** that reads from or writes to a removable, nonvolatile optical disk **162** such as a CD-ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive **144**, and magnetic disk drive **156** and optical disk drive **160** are typically connected to the system bus **136** by a nonvolatile memory interface, such as interface **166**.

[0107] The drives or other mass storage devices and their associated computer storage media discussed above and

illustrated in FIG. 7, provide storage of computer readable instructions, data structures, program modules and other data for the computer 130. In FIG. 7, for example, hard disk drive 154 is illustrated as storing operating system 170, application programs 172, other program modules 174, and program data 176. Note that these components can either be the same as or different from operating system 144, application programs 146, other program modules 148, and program data 150. Operating system 170, application programs 172, other program modules 174, and program data 176 are given different numbers here to illustrate that, at a minimum, they are different copies.

[0108] A user may enter commands and information into computer 130 through input devices or user interface selection devices such as a keyboard 180 and a pointing device 182 (e.g., a mouse, trackball, pen, or touch pad). Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to processing unit 132 through a user input interface 184 that is coupled to system bus 136, but may be connected by other interface and bus structures, such as a parallel port, game port, or a Universal Serial Bus (USB). A monitor 188 or other type of display device is also connected to system bus 136 via an interface, such as a video interface 190. In addition to the monitor 188, computers often include other peripheral output devices (not shown) such as a printer and speakers, which may be connected through an output peripheral interface (not shown).

[0109] The computer 130 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 194. The remote computer 194 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 130. The logical connections depicted in FIG. 7 include a local area network (LAN) 196 and a wide area network (WAN) 198, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and global computer networks (e.g., the Internet).

[0110] When used in a local area networking environment, computer 130 is connected to the LAN 196 through a network interface or adapter 186. When used in a wide area networking environment, computer 130 typically includes a modem 178 or other means for establishing communications over the WAN 198, such as the Internet. The modem 178, which may be internal or external, is connected to system bus 136 via the user input interface 194, or other appropriate mechanism. In a networked environment, program modules depicted relative to computer 130, or portions thereof, may be stored in a remote memory storage device (not shown). By way of example, and not limitation, FIG. 7 illustrates remote application programs 192 as residing on the memory device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0111] Generally, the data processors of computer 130 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs.

From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the steps described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described herein.

[0112] For purposes of illustration, programs and other executable program components, such as the operating system, are illustrated herein as discrete blocks. It is recognized, however, that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

[0113] Although described in connection with an exemplary computing system environment, including computer 130, the invention is operational with numerous other general purpose or special purpose computing system environments or configurations. The computing system environment is not intended to suggest any limitation as to the scope of use or functionality of the invention. Moreover, the computing system environment should not be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0114] The invention may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include, but are not limited to, routines, programs, objects, components, and data structures that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0115] The following scenarios illustrate operation of the software of the invention.

[0116] On-Demand Content

[0117] In a server 108 to client 110 network interruption scenario, one or more clients such as clients 110 viewing on-demand content have their network connection interrupted. Automatic reconnect logic 112, 114 minimizes the impact to each viewer affected by the temporary network outage. The reconnect logic 112, 114 allows the client 110 to restart at the point the connection was lost by seeking to that point in the file upon successfully reconnecting to the server 108. If the content is not seekable, the program element shall be restarted at the beginning.

[0118] In a source to server network interruption scenario, all clients **110** that are streaming on-demand content obtained from another location by the edge server **108** will be affected. Automatic reconnect logic **112**, **114** minimizes the impact to all viewers affected by the temporary network outage. The reconnect logic **112**, **114** allows the client **110** to restart at the point the connection was lost by seeking to that point in the file upon successfully reconnecting to the server **108**. If the content is not seekable, the program element shall be restarted at the beginning.

[0119] Broadcast Content

[0120] A source to server network interruption scenario is routinely encountered by large CDNs. In this scenario, all clients **110** that are streaming content obtained from another location by the edge server **108** are affected. If the source content is live, the customer may experience a gap in the program even when automatic reconnect logic **112**, **114** is successful. However, automatic reconnect logic **112**, **114** minimizes the impact to all viewers affected by the temporary network outage.

[0121] In a server **108** to client **110** network interruption scenario, one or more clients such as clients **110** viewing broadcast content have their network connection interrupted. Due to the nature of a broadcast, the customer experiences a gap in the program even when automatic reconnect logic **112**, **114** is successful. However, automatic reconnect logic **112**, **114** minimizes the impact to the specific viewer(s) affected by the temporary network outage.

[0122] The following examples illustrate specific embodiments of the invention.

[0123] Content Distribution Network Scenario

[0124] Some CDNs have complicated distribution scenarios involving combinations of origin and distribution servers such as server **108** using the Internet for some of their distribution feeds. When temporary problems on the Internet result in the distribution connection being severed, all downstream clients **110** that are streaming the content are disconnected. This results in the loss of thousands of clients **110** (and subsequent lost revenue opportunities often dependent upon on successful usage logging statistics) when a network feed is temporarily interrupted.

[0125] The automatic client reconnection software reduces the scenarios where clients **110** are dropped due to distribution network interruptions. For example, some platforms shall support a temporary distribution network outage of at least 90 seconds before clients **110** are terminated by the servers **108** downstream from the distribution network interruption. Furthermore, assuming the reconnection attempt is successful, the logging usage information for clients **110** is complete. Lost revenue due to network problems will be reduced.

[0126] Listening to an Internet Radio Station All Day

[0127] In one example, a user loves to listen to an Internet sports radio station all day at work while working on a

computer. Unfortunately, the LAN is notoriously unreliable (e.g., routers are often rebooted). In addition, the firewall often times out TCP connections and resets them. The ISP is also unreliable. Network interruptions often exceed 10 seconds. As a result, the user often gets disconnected from the Internet radio station server, and an annoying dialog pops up forcing a manual reconnect. Sometimes, the user has to try a few times before reconnecting back to the Internet radio.

[0128] The automatic reconnect software of the invention addresses the problem the user is currently experiencing. The player employs software to attempt to reconnect multiple times before popping up an error dialog. A configuration option in the player allows the user to set the number of attempts. With the invention, the user is able to leave the player running indefinitely. Movie Scenario

[0129] In another example, the user recently subscribed to a video-on-demand trial in an assisted-living apartment. The user typically watches 2-4 action movies per week with friends. When the user orders a new movie, the CDN precedes the start of the movie with trailers for other action movies that the user might be interested in. Because the CDN mixes and matches these trailers with other customers, the trailers are separate files (e.g., advanced streaming format files). The trailers and movie are tied together sequentially by using a server-side playlist dynamically generated in response to the movie order.

[0130] The user has a cable modem connection that is susceptible to occasional temporary outages. Sometimes, while watching movies, the temporary network outage causes the TCP connection to be reset or the starvation timer on the client **110** to fire. With the reconnect software of the invention, the user only experiences a pause in the playback of the movie. The user's player does not display an error requiring user intervention. The user does not lose the connection or the location in the server-side playlist. As such, the user does not need to search through a server-side playlist or view error messages. The user simply views the movie without noticing any of the network outages.

[0131] When introducing elements of the present invention or the embodiment(s) thereof, the articles "a," "an," "the," and "said" are intended to mean that there are one or more of the elements. The terms "comprising," "including," and "having" are intended to be inclusive and mean that there may be additional elements other than the listed elements.

[0132] In view of the above, it will be seen that the several objects of the invention are achieved and other advantageous results attained.

[0133] As various changes could be made in the above constructions, products, and methods without departing from the scope of the invention, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

Appendix A

[0094] When the client 110 experiences a network outage, it will attempt to automatically reconnect to the same session when the connection is reestablished, as shown in the following network trace.

```
RTSP/1.0 200 OK
Content-Type: application/sdp
Vary: Accept
X-Playlist-Gen-Id: 2
Content-Length: 2781
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 1
Timestamp: 1 0.0
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-revalidate
Etag: "2139795"
```

```
SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/rtx RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2576-2577;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 2
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *;q=0.1
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1
```

RTSP/1.0 200 OK
Transport: RTP/AVP/UDP;unicast;server_port=6002-6003;client_port=2576-2577;mode=PLAY
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 2
Timestamp: 1 0.0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-revalidate
Etag: "2139795"
X-Feedback-Stream-Id: 35129

SET_PARAMETER rtsp://MyServer.MyDomain.com/welcome2.asf RTSP/1.0
Content-Type: application/x-rtsp-udp-packetpair;charset=UTF-8
Content-Length: 29
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 3
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 200 OK
Content-Type: application/x-rtsp-udp-packetpair;charset=UTF-8
Content-Length: 29
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 3
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/audio RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2578;ssrc=72b2ac0d;mode=PLAY,
RTP/AVP/TCP;unicast;interleaved=0-1;ssrc=72b2ac0d;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 4
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
X-Batch: 1:2
Timestamp: 1

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5 RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2578;ssrc=41740f9d;mode=PLAY,
RTP/AVP/TCP;unicast;interleaved=2-3;ssrc=41740f9d;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 5
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
X-Batch: 2:2
Timestamp: 1

RTSP/1.0 200 OK
Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2579;client_port=2578;s
src=8ce7466f;mode=PLAY
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 4
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-
revalidate
Etag: "2139795"

RTSP/1.0 200 OK
Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2579;client_port=2578;s
src=bc93c310;mode=PLAY
Date: Fri, 04 May 2001 01:21:57 GMT
CSeq: 5
Timestamp: 1 0.0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-
revalidate
Etag: "2139795"

PLAY rtsp://MyServer.MyDomain.com/welcome2.asf RTSP/1.0
Bandwidth: 2147483647
X-Accelerate-Streaming: AccelDuration=10000;AccelBandwidth=891289
X-Playlist-Gen-Id: 2
Date: Fri, 04 May 2001 01:22:00 GMT
CSeq: 6
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Scale: 1
Range: npt=0.000-
Timestamp: 1

RTSP/1.0 200 OK
Date: Fri, 04 May 2001 01:22:00 GMT
CSeq: 6
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
Range: npt=0.000-70.891
Scale: 1
X-Accelerate-Streaming: AccelBandwidth=891289;AccelDuration=10000
Speed: 1.000
RTP-Info:
url="rtsp://MyServer.MyDomain.com/welcome2.asf/audio";seq=65450;rtptime=0,
url="rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5";seq=65343;rtp
time=0

SET_PARAMETER rtsp://MyServer.MyDomain.com/welcome2.asf RTSP/1.0
Content-Type: application/x-Logconnectstats;charset=UTF-8
Content-Length: 171
Date: Fri, 04 May 2001 01:22:00 GMT
CSeq: 7
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *;q=0.1
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 200 OK
Date: Fri, 04 May 2001 01:22:00 GMT
CSeq: 7
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201

Network Outage

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/audio RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2578;ssrc=72b2ae0d;mode=PLAY,
RTP/AVP/UDP;unicast;client_port=2578;ssrc=72b2ae0d;m
ode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:22:17 GMT
CSeq: 1
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5 RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2578;ssrc=41740f9d;mode=PLAY,
RTP/AVP/UDP;unicast;client_port=2578;ssrc=41740f9d;m
ode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:22:17 GMT
CSeq: 2
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/rtx RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2576-2577;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:22:17 GMT
CSeq: 3
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 200 OK

Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2579;client_port=2578;s
src=8ce7466f;mode=PLAY
Date: Fri, 04 May 2001 01:22:17 GMT
CSeq: 1
Timestamp: 1 0.0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201

RTSP/1.0 200 OK

Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2579;client_port=2578;s
src=bc93c310;mode=PLAY
Date: Fri, 04 May 2001 01:22:19 GMT
CSeq: 2
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201

RTSP/1.0 200 OK

Transport: RTP/AVP/UDP;unicast;server_port=6002-6003;client_port=2576-
2577;ssrc=6a1c5f43;mode=PLAY
Date: Fri, 04 May 2001 01:22:19 GMT
CSeq: 3
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
X-Feedback-Stream-Id: 35129

PLAY rtsp://MyServer.MyDomain.com/welcome2.asf RTSP/1.0
Bandwidth: 2147483647
X-Accelerate-Streaming: AccelDuration=18000;AccelBandwidth=891289
X-Playlist-Gen-Id: 2
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:22:19 GMT
CSeq: 4
Session: 2584499203
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Scale: 1
Range: npt=17.400-
Timestamp: 1

RTSP/1.0 200 OK
Date: Fri, 04 May 2001 01:22:19 GMT
CSeq: 4
Timestamp: 1 0
Session: 2584499203;timeout=60
Server: Server/9.0.0.201
Range: npt=17.400-70.891
Scale: 1
X-Accelerate-Streaming: AccelBandwidth=891289;AccelDuration=18000
Speed: 1.000
RTP-Info:
url="rtsp://MyServer.MyDomain.com/welcome2.asf/audio";seq=79;rtptime=17400,
url="rtsp://MyServer.MyDomain.com/welcome2.asf/rtx";seq=65396;rtptime
=17400,
url="rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5";seq=65343;rtptime=1
7400

Appendix B

[0095] When the client 110 attempts to automatically reconnect to the same session after a network outage, the session may have expired. In this case, the client 110 makes a new attempt to connect, this time without including the session ID. The server 108 creates a new session and returns the ID, as shown in the following network trace.

Network Outage

```
SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/audio RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2544;ssrc=3c273be8;mode=PLAY,
RTP/AVP/UDP;unicast;client_port=2544;ssrc=3c273be8;m
ode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 1
Session: 3243206664
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1
```

```
SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5 RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2544;ssrc=0f1d8df1;mode=PLAY,
RTP/AVP/UDP;unicast;client_port=2544;ssrc=0f1d8df1;m
ode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 2
Session: 3243206664
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1
```

SETUP rtsp://MyServer.MyDomain.com/welcomc2.asf/rtx RTSP/1.0
X-Playlist-Gen-Id: 2
Transport: RTP/AVP/UDP;unicast;client_port=2542-2543;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 3
Session: 3243206664
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *;q=0.1
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 454 Session Not Found
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 1
Timestamp: 1 0
Server: Server/9.0.0.201

RTSP/1.0 454 Session Not Found
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 2
Timestamp: 1 0
Server: Server/9.0.0.201

RTSP/1.0 454 Session Not Found
Date: Fri, 04 May 2001 01:17:10 GMT
CSeq: 3
Timestamp: 1 0
Server: Server/9.0.0.201

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/audio RTSP/1.0
X-Playlist-Gen-Id:
Transport: RTP/AVP/UDP;unicast;client_port=2546;ssrc=3c273be8;mode=PLAY,
RTP/AVP/TCP;unicast;interleaved=0-1;ssrc=3c273be8;mo
de=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 4
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *;q=0.1
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 200 OK
Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2563;client_port=2546;s
src=dabe4914;mode=PLAY
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 4
Timestamp: 1 0.0
Session: 3828318172;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-
revalidate
Etag: "2139795"

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/stream=5 RTSP/1.0
X-Playlist-Gen-Id:
Transport: RTP/AVP/UDP;unicast;client_port=2546;ssrc=0f1d8df1;mode=PLAY,
RTP/AVP/TCP;unicast;interleaved=0-1;ssrc=0f1d8df1;mo
de=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 5
Session: 3828318172
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-
30C2D12D7C8B
Accept-Language: en-us, *;q=0.1
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

SETUP rtsp://MyServer.MyDomain.com/welcome2.asf/rtx RTSP/1.0
X-Playlist-Gen-Id:
Transport: RTP/AVP/UDP;unicast;client_port=2548-2549;mode=PLAY
If-Match: "{83A04BD0-FD30-1984-4994-0A22CA116ED3}"
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 6
Session: 3828318172
User-Agent: Player/9.00.00.0201 guid/AD260B67-5E90-4E96-AED0-30C2D12D7C8B
Accept-Language: en-us, *,q=0.1
Accept-Charset: UTF-8, *,q=0.1
X-Accept-Authentication: NTLM, Digest, Basic
Timestamp: 1

RTSP/1.0 200 OK
Transport:
RTP/AVP/UDP;unicast;source=172.29.232.226;server_port=2563;client_port=2546;s
src=2b6c9bcf;mode=PLAY
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 5
Timestamp: 1 0.0
Session: 3828318172;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-
revalidate
Etag: "2139795"

RTSP/1.0 200 OK
Transport: RTP/AVP/UDP;unicast;server_port=6002-6003;client_port=2548-
2549;ssrc=9b973387;mode=PLAY
Date: Fri, 04 May 2001 01:17:12 GMT
CSeq: 6
Timestamp: 1 0
Session: 3828318172;timeout=60
Server: Server/9.0.0.201
Last-Modified: Wed, 06 Dec 2000 15:38:32 GMT
Cache-Control: content-size="2139795", max-age=86399, must-revalidate, proxy-
revalidate
Etag: "2139795"
X-Feedback-Stream-Id: 24812

Appendix C

[0096] The client 110 sends out connection related statistics to the server 108 on a regular basis using the RTSP SET_PARAMETER method in a unicast session. The data in the logging information record is in the form of an XML description (see example below).

[0097] The various statistical parameters that remain constant throughout the session are sent only once at the beginning of the session. The other dynamically changing parameters are sent regularly, the frequency of reporting set by the statistics reporting interval parameter sent in the initial request.

[0098] The following example shows a network trace of statistics reporting. The client 110 sends the following information to the server 108.

```
SET_PARAMETER rtsp://test.com/eagles/foo RTSP/1.0
CSeq: 431
Content-Type: application/x-logparameters
Session: 12345
Content-Length: 21 //length of log data
c-pkts_received: 1000
c-resendreqs: 15
:
: //log data
```

In response to the client 110, the server 108 sends the following information.

```
RTSP/1.0 200 OK
CSeq: 431
Session: 12345;timeout=10
```

Statistics Parameters

[0099] The following table describes exemplary statistics that may be sent from the client 110 to the server 108 for logging purposes. Some of the statistics are static parameters (i.e., sent once in the beginning or at the end of the session).

Table C1: Logging Fields

| Field | Meaning | Sample data |
|--------------|---|---|
| c-ip | Client IP address | 157.56.81.76 |
| date | Date when Client connected to station | 1998-01-09 |
| Time | Time when Client connected to station | 05:36:03 |
| c-dns | Client computer. Note, due to privacy concerns the client sends up an empty value (denoted as a dash character) | MyServer.MyDomain.com |
| cs-uri-stem | The URI stem of the content that was requested. | /videos/MyHomeMovie.wsv |
| c-starttime | The timestamp (in seconds, no fractions) when you played the file in normal mode | 3285 |
| c-endtime | The timestamp (in seconds, no fractions) when the Client finished playing the file in normal mode | 10345 |
| x-duration | How long you tried to receive the stream (in seconds) | 0 |
| Avgbandwidth | Average bandwidth (in bytes) at which the Client was connected to the Server | 21429 |
| protocol | The protocol used to access the stream | Asfu |
| c-playerid | GUID | {c579d042-cecc-11d1-bb31c-00a0c9603954} |

| | | |
|------------------|---|----------------------------|
| c-playerversion | The player version number | 5.1.51.413 |
| c-playerlanguage | The language of the Client. This is a code for the country | enu |
| cs(User-Agent) | If the player was embedded in a browser, this field refers to the browser type that was used | - |
| cs(Referer) | URL to the Web page that the player was embedded within (if the player was embedded) | - |
| c-hostexe | The host application, for example, a Web page in a browser, an applet or stand-alone media player | XYZplayer.exe |
| c-hostexeversion | Version number of the host application | 5.1.5.413 |
| c-os | Client computer's operating system | XYZos |
| c-osversion | Version # of the Client's OS | 3.0.0.111 |
| audiocodec | Audio codec used in stream | Compression_Layer-3 |
| vidcocodec | Video codec used to encode the stream | Compression_Video_Codec_V2 |
| c-cpu | Client computer's CPU | XYZProcessor |
| c-status | Codes that describe the Client's status. Mapped to http/rtsp status codes; 200 is success, 404 is file not found. | 200 |
| c-bytes | Number of bytes received by the Client from the Server. C-bytes and sc-bytes | 28583 |

| | | |
|-------------------------|--|-----|
| | should be the same, if not, there was packet loss. | |
| c-pkts-received | Number of packets received by the Client | 0 |
| c-resendreqs | The number of requests made by the Client to receive new packets | 0 |
| c-pkts-recovered-ECC | Number of packets that were repaired and recovered on the Client layer | 0 |
| c-pkts-recovered-resent | Number of packets that were recovered because they were resent via UDP | 1 |
| c-buffercount | Number of times the Client buffered while playing the stream | 6 |
| c-totalbuffertime | The total time (in seconds) the Client used in buffering the stream | 100 |
| c-quality | The measure of how well the stream was received (on a scale from 0-100%) | 100 |

Example Logging Record

[00100] The following is an example logging information record.

Protocol Headers:

SET_PARAMETER rtsp://MyServer.MyDomain.com/welcome2.asf RTSP/1.0..

Content-Type: application/x-Logplaystats;charset=UTF-8..

Content-Length: 1597..Date: Wed, 14 Mar 2001 17:47:35 GMT..

CSeq: 9..Session: 885575573..

User-Agent: Player/9.0.0.189

guid/94a7f5df-704a-4ee8-a125-622e5c68f00f..

Accept-Language: en-us, *,q=0.1..

Accept-Charset: UTF-8, *,q=0.1..

X-Accept-Authentication: NTLM, Digest, Basic

Timestamp: 2

Protocol Body:

```
<XML>.<Summary>0.0.0.0 2001-03-14 17:47:35-
rtsp://MyServer.MyDomain.com/welcome2.asf 010 1 200 {94a7f5df-704a-4ee8-a125-
622e5c68f00f} 9.0.0.179 en-US - - XYZplayer.exe 7.0.0.1958 XYZos 5.0.0.2195
XYZProcessor 68 0192000 rtsp UDP Compression_Audio_Codec_V2
Compression_Video_Codec_V3 - - 2400 81 - 140 0 0 0 0 0 1 1 100 - - - rtsp://test
/welcome2.asf </Summary>.
```

<c-pkts-received>140</c-pkts-received>.

<c-pkts-lost -client>0 </c-pkts-lost-client>

<c-pkts-lost-net>0</c-pkts-lost-net>.

<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>.

<c-resendreqs>0</c-resendreqs>

<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>

<c-pkts-recovered-resent>0</c-pkts-recovered-resent>

<c-buffercount>1</c-buffercount>

<c-totalbuffertime>1</c-totalbuffertime>.

<c-quality>100</c-quality>.

<filelength>68</filelength >.

<filesize >0</filesize>.

<audiocodec> Compression_Audio_Codec_V2</audiocodec>.

```
<videocodec> Compression_Video_Codec_V3</videocodec>.
<c-playerversion>9.0.0.1</c-playervers >.
<c-playerlanguage >en-US</c-playerlanguage>.
<c-hostexe>XYZplayer.exe</c-hostexe>.
<c-hostexever >7.0.0.19.58</c-hostexever>.
<cs-url>rtsp://MyServer.MyDomain.com/welcome2.asf</csurl>.
<ContentDescription>.
<CONTENT_DESCRIPTION_TITLE>Media</CONTENT_DESCRIPTION_TI
TLE>
<Copied MetaData From Playlist File>1</Copied MetaData From PlaylistFile>
<Rating></Rating>.
<CONTENT_DESCRIPTION_DESCRIPTION></CONTENT_DESCRIPTION_
DESCRIPTION>.
<CONTENT_DESCRIPTION_COPYRIGHT >(c)
1999</CONTENT_DESCRIPTION_COPYRIGHT>.
<CONTENT_DESCRIPTION_AUTHOR>Media</CONTENT_DESCRIPTION
_AUTHOR>.
</ContentDescription >
</XML>
```

What is claimed is:

1. A method of streaming media content from a server to at least one client, said method comprising:

establishing a streaming media connection between the server and the at least one client;

streaming the media content from the server to the client;

receiving, by the client, the streamed media content from the server;

sending a reconnect request from the client to the server if said streaming is interrupted;

receiving, by the server, the reconnect request from the client;

re-establishing the streaming media connection with the client; and

continuing with said streaming the media content and said receiving the streamed media content.

2. The method of claim 1, wherein said sending comprises sending a reconnect request having a stream identifier and a session identifier.

3. The method of claim 1, wherein said streaming comprises maintaining a state of the client.

4. The method of claim 3, wherein said maintaining comprises maintaining the state of the client for a preset time period after said streaming is interrupted.

5. The method of claim 4, wherein said re-establishing comprises associating the maintained state with the client sending the reconnect request.

6. The method of claim 1, wherein said streaming comprises streaming the media content to the client from a file system accessible by the server.

7. The method of claim 1, wherein said streaming comprises streaming, by the server, the media content to the client from another server.

8. The method of claim 1, further comprising transmitting state information from the client to the server.

9. The method of claim 1, wherein said streaming comprises streaming the media content from the server to the client via a real-time streaming protocol.

10. The method of claim 1, wherein said streaming comprises streaming the media content from the server to the client via a hypertext transfer protocol.

11. The method of claim 1, wherein one or more computer-readable media have computer-executable instructions for performing the method of claim 1.

12. A method of streaming media content to at least one client, said method comprising:

establishing a streaming media connection with at least one client;

streaming the media content to the client;

receiving a reconnect request from the client if said streaming is interrupted;

re-establishing the streaming media connection with the client; and

continuing with said streaming the media content.

13. The method of claim 12, wherein said streaming comprises maintaining a state of the client.

14. The method of claim 13, wherein said maintaining comprises maintaining the state of the client for a preset time period after said streaming is interrupted.

15. The method of claim 14, wherein said maintaining comprises deleting the state of the client if the preset time period has elapsed.

16. The method of claim 15, wherein said maintaining further comprises logging an error.

17. The method of claim 12, wherein one or more computer-readable media have computer-executable instructions for performing the method of claim 12.

18. The method of claim 12, wherein said receiving comprises receiving a reconnect request having a stream identifier and a session identifier.

19. The method of claim 18, wherein said streaming comprises maintaining a state of the client in a state repository, and wherein said re-establishing comprises searching for the received session identifier in the state repository.

20. The method of claim 19, wherein if the received session identifier is not found within the state repository, said re-establishing further comprises establishing another streaming media connection with the client, and said continuing comprises streaming the media content associated with the received stream identifier to the client.

21. The method of claim 19, wherein if the received session identifier is found within the state repository, said re-establishing further comprises searching for the received stream identifier within the state repository.

22. The method of claim 21, wherein if the stream identifier is found within the state repository, said continuing comprises streaming the media content associated with the received stream identifier to the client.

23. The method of claim 21, wherein if the received stream identifier is not found within the state repository, said re-establishing further comprises transmitting one or more other stream identifiers to the client for selection by the client.

24. The method of claim 23, wherein said re-establishing further comprises receiving a playback request from the client in response to said transmitting, said playback request comprising at least one of the other stream identifiers.

25. The method of claim 24, wherein said continuing comprises streaming the media content associated with the received at least one of the other stream identifiers.

26. A method of receiving media content streamed from a server, said method comprising:

establishing a streaming media connection with the server;

receiving the media content streamed from the server;

transmitting a reconnect request to the server if said receiving is interrupted;

re-establishing the streaming media connection with the server; and

continuing with said receiving the streamed media content.

27. The method of claim 26, wherein said transmitting further comprises:

delaying said transmitting for a preset time period; and

incrementing a reconnect counter.

28. The method of claim 27, further comprising repeating said transmitting until said re-establishing occurs or until the reconnect counter exceeds a threshold.

29. The method of claim 28, wherein said re-establishing further comprises resetting the reconnect counter.

30. The method of claim 26, wherein said re-establishing further comprises transmitting state data to the server.

31. The method of claim 26, wherein one or more computer-readable media have computer-executable instructions for performing the method of claim 26.

32. The method of claim 26, further comprising sending state data to the server to be maintained in a state repository.

33. The method of claim 32, wherein said transmitting comprises transmitting a reconnect request to the server having a session identifier and a stream identifier.

34. The method of claim 33, further comprising receiving a message from the server indicating that the transmitted session identifier was not found within the state repository, wherein said re-establishing comprises establishing another streaming media connection with the server, and said continuing comprises receiving the media content associated with the transmitted stream identifier from the server.

35. The method of claim 33, further comprising receiving a message from the server indicating that the transmitted session identifier and the transmitted stream identifier have been found within the state repository, wherein said continuing comprises receiving the media content associated with the transmitted stream identifier from the server.

36. The method of claim 33, further comprising receiving a message from the server indicating that the transmitted session identifier was found in the state repository and the transmitted stream identifier was not found within the state repository, wherein said re-establishing further comprises receiving one or more other stream identifiers from the server.

37. The method of claim 36, further comprising selecting at least one of the other stream identifiers, and wherein said re-establishing further comprises transmitting a playback request to the server, said playback request comprising the selected, other stream identifiers.

38. The method of claim 37, wherein said continuing comprises receiving from the server the media content associated with the selected, other stream identifiers.

39. In a system wherein a server streams media content to at least one client, one or more computer-readable media having computer-executable components comprising:

a server component; and

at least one client component, wherein the server component and the client component comprise computer-executable instructions for exchanging one or more messages to re-map the state of the client and to re-synchronize playback of the content if the streaming is interrupted.

40. The computer-readable media of claim 39, wherein the server component comprises computer-executable instructions for:

establishing a streaming media connection with at least one client;

streaming the media content to the client;

receiving a reconnect request from the client if said streaming is interrupted;

re-establishing the streaming media connection with the client; and

continuing with said streaming the media content.

41. The computer-readable media of claim 39, wherein the at least one client component comprises computer-executable instructions for:

establishing a streaming media connection with the server;

receiving the media content streamed from the server;

transmitting a reconnect request to the server if said receiving is interrupted;

re-establishing the streaming media connection with the server; and

continuing with said receiving the streamed media content.

42. One or more computer-readable media having stored thereon a data structure representing a reconnect request transmitted by a client to a server to re-establish an interrupted streaming media session, said data structure comprising:

a session identifier identifying the interrupted streaming media session;

a stream identifier identifying a media stream streamed by the server to the client in the interrupted streaming media session.

* * * * *

APPENDIX B-4



US 20040133467A1

(19) **United States**

(12) **Patent Application Publication**
Siler

(10) **Pub. No.: US 2004/0133467 A1**

(43) **Pub. Date: Jul. 8, 2004**

(54) **METHOD AND APPARATUS FOR
SELECTING STREAMING MEDIA IN
REAL-TIME**

ation-in-part of application No. 09/625,443, filed on
Jul. 26, 2000, now abandoned.

(76) Inventor: **Gregory Aaron Siler, Garland, TX
(US)**

Publication Classification

Correspondence Address:
**ROBERTS ABOKHAIR & MARDULA
SUITE 1000
11800 SUNRISE VALLEY DRIVE
RESTON, VA 20191 (US)**

(51) **Int. Cl.⁷ G06F 17/60**

(52) **U.S. Cl. 705/14; 709/218**

(21) Appl. No.: **10/463,120**

(22) Filed: **Jun. 17, 2003**

Related U.S. Application Data

(63) Continuation of application No. 09/642,037, filed on
Aug. 18, 2000, now abandoned, which is a continu-

(57) **ABSTRACT**

Streaming media over a packet switched network includes
processes for tracking which users are receiving a particular
media stream and how long each of the users receives in
order to collect time line information. Advertisements are
preferably selected in real-time based on predefined criteria
and switched, in response to a trigger, in place of a source
signal during a streaming session.

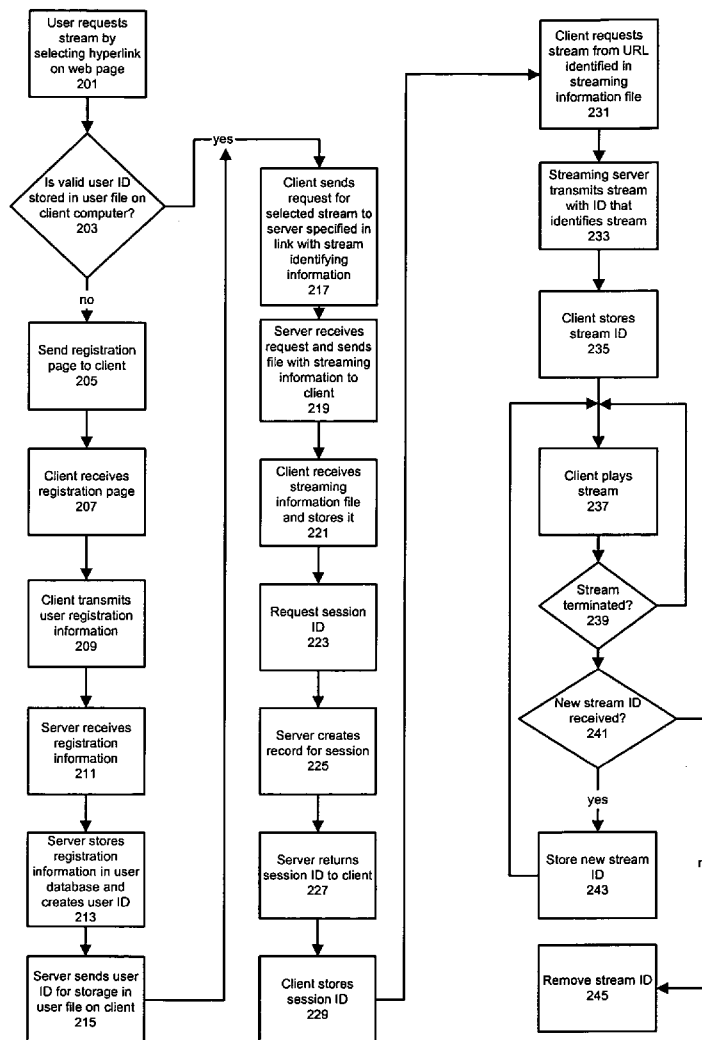


FIG. 1A

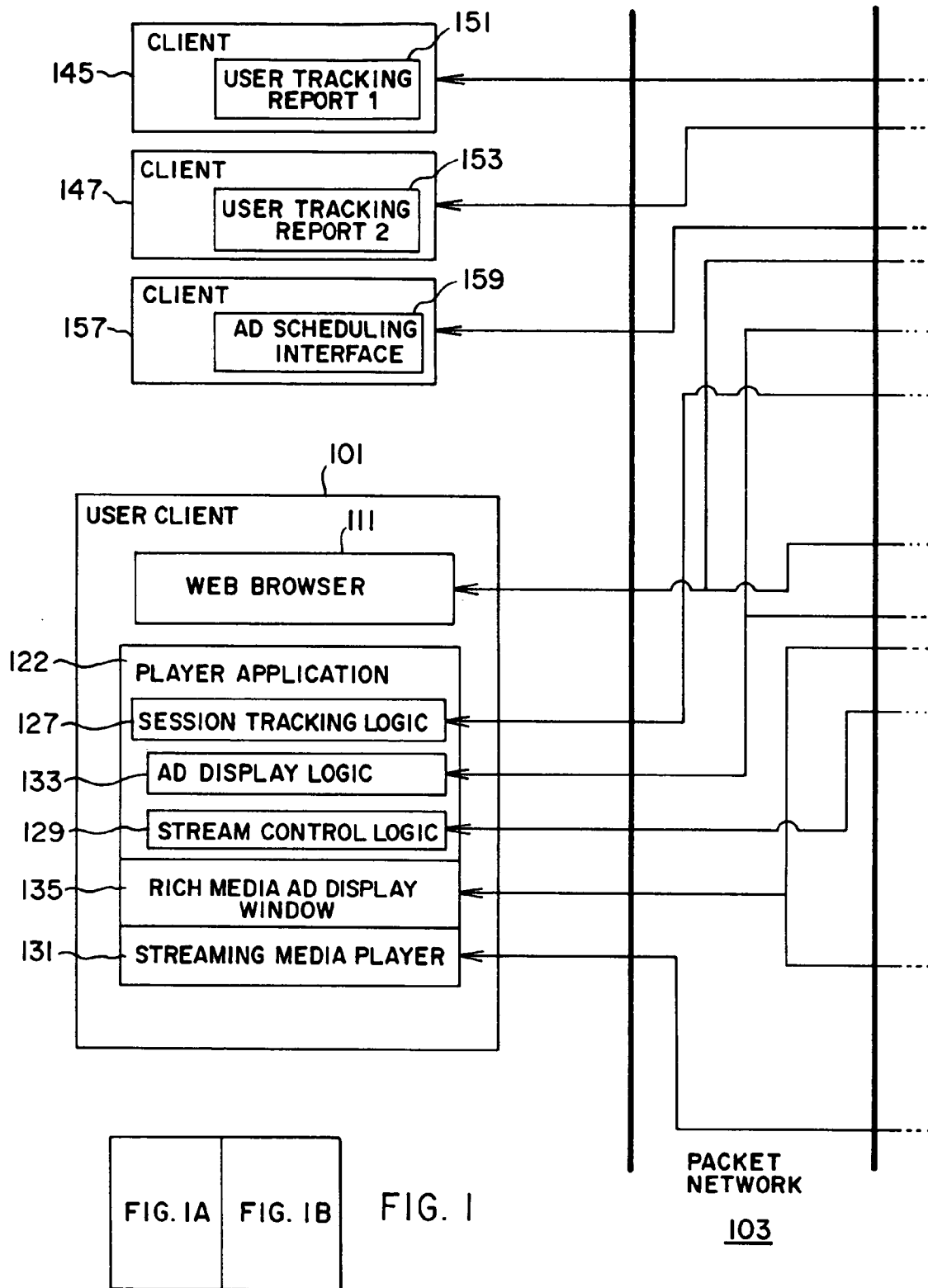


FIG. 1B

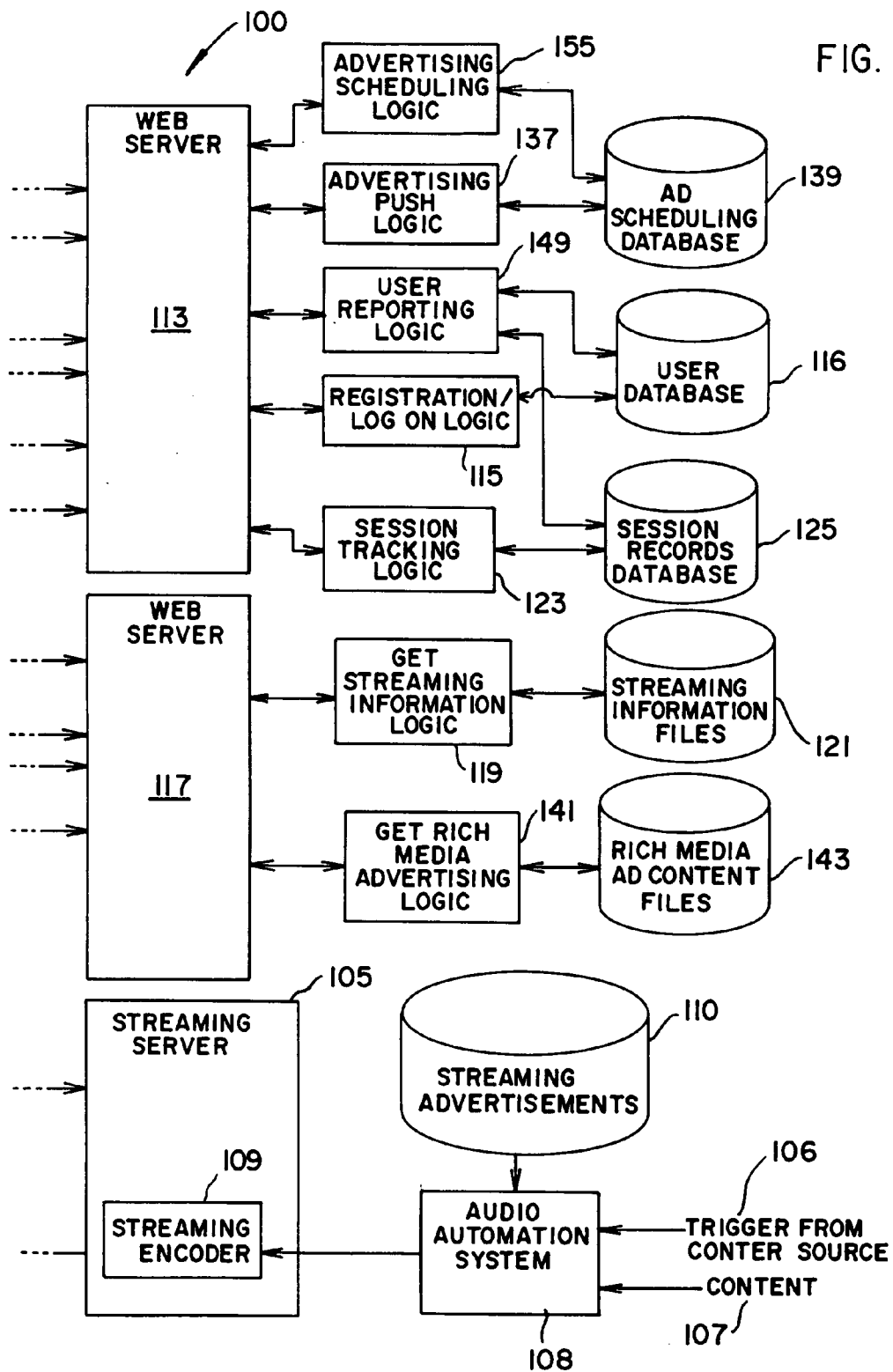
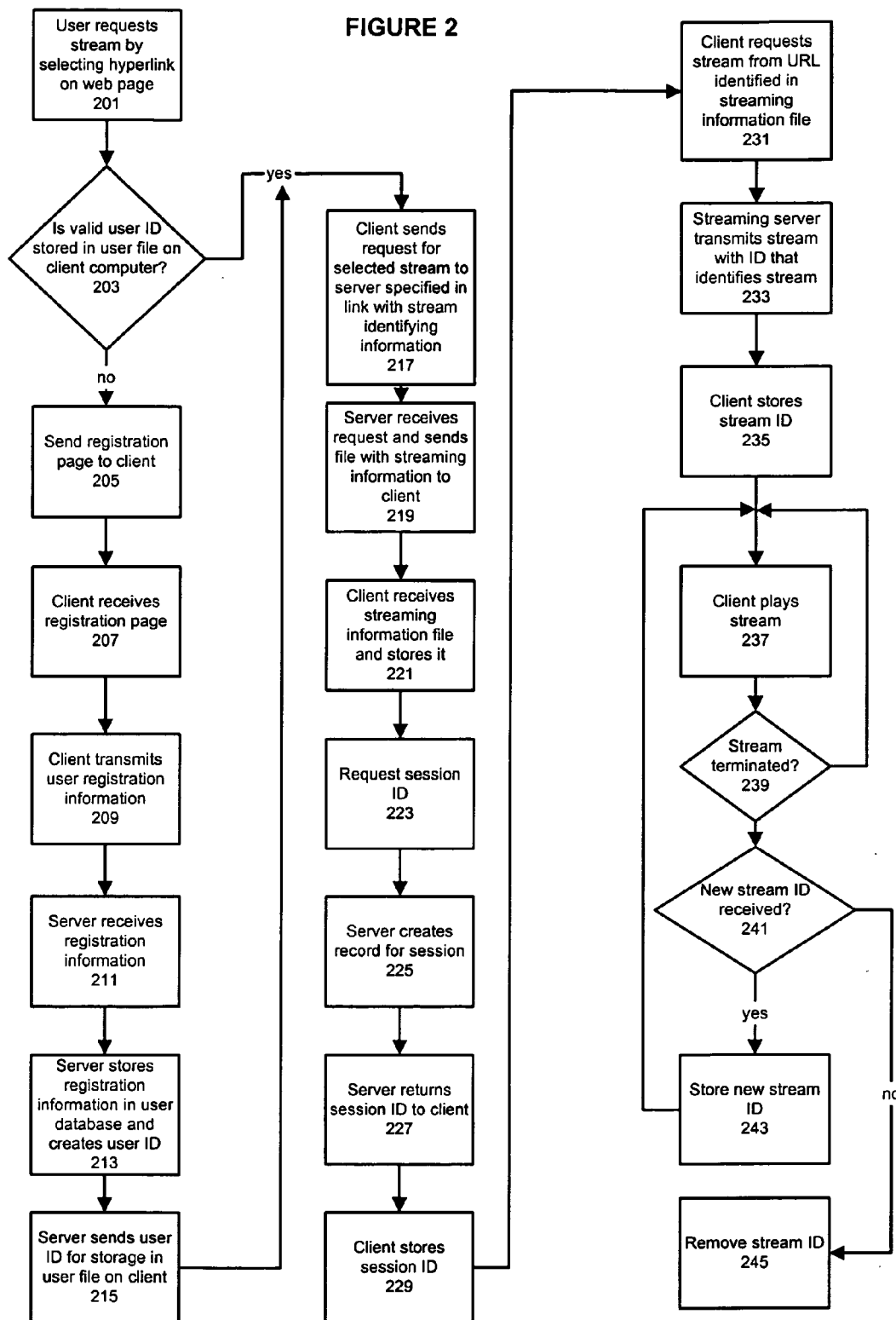


FIGURE 2



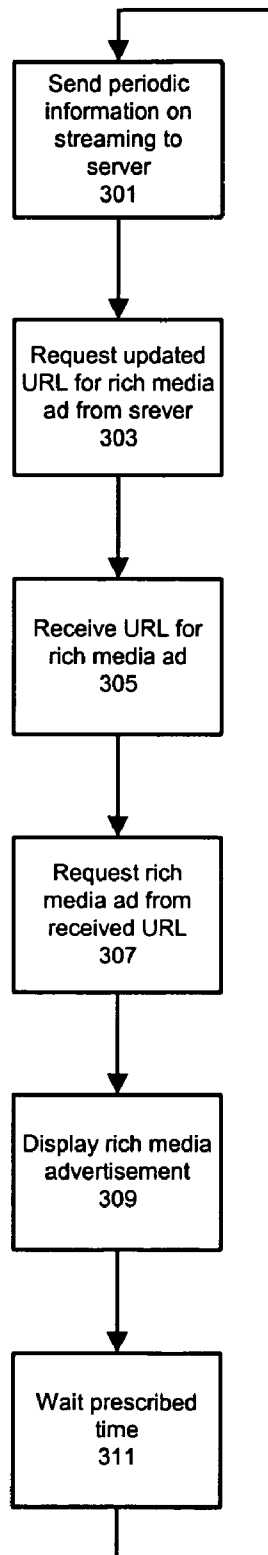


FIGURE 3

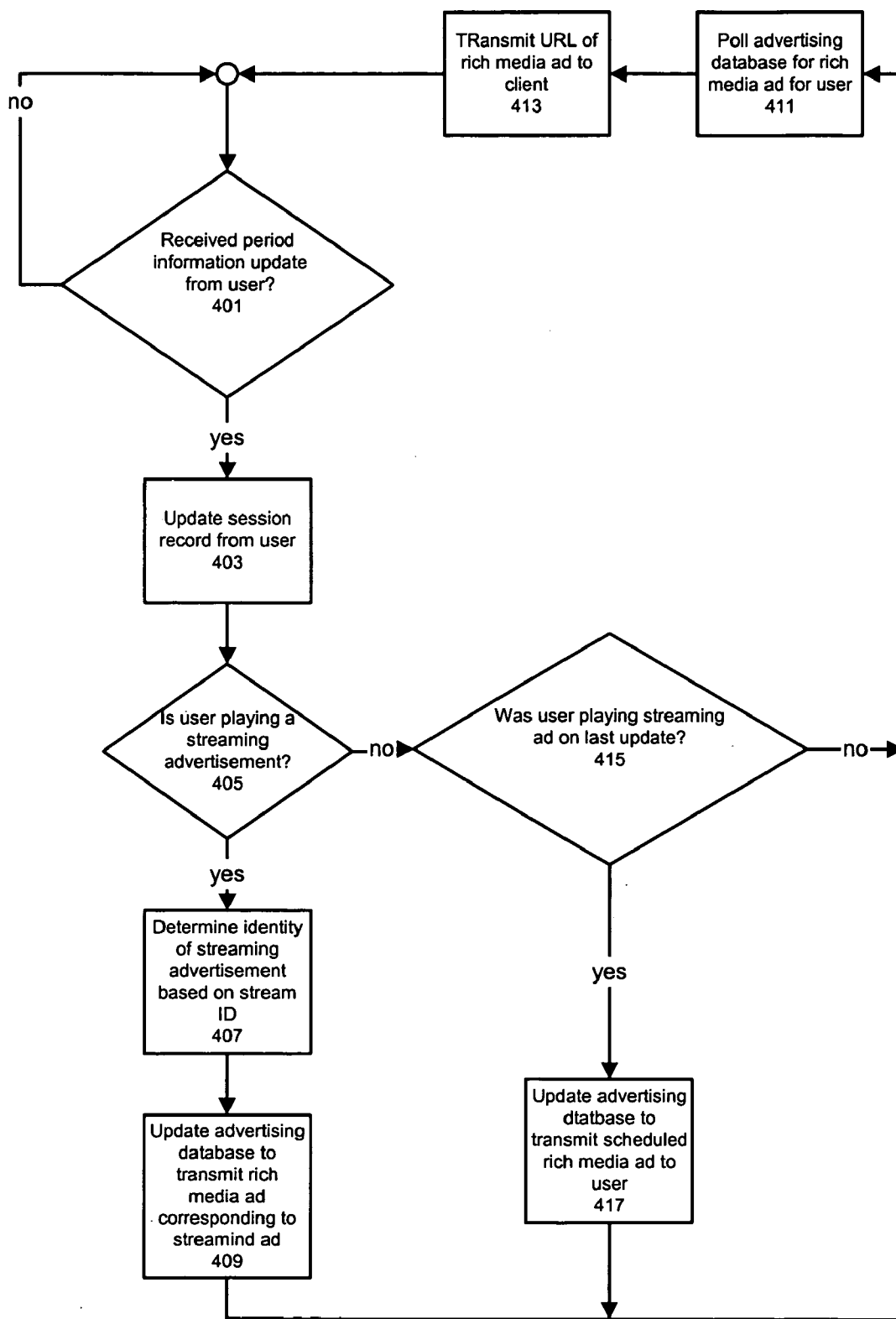


Figure 4

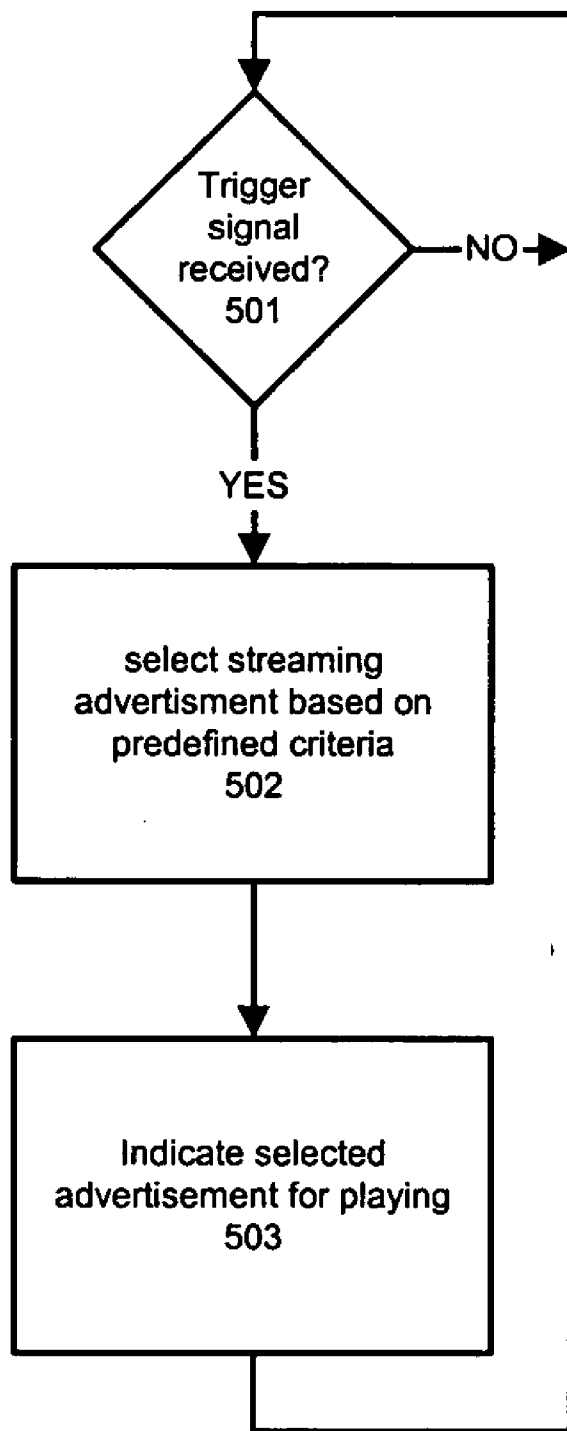
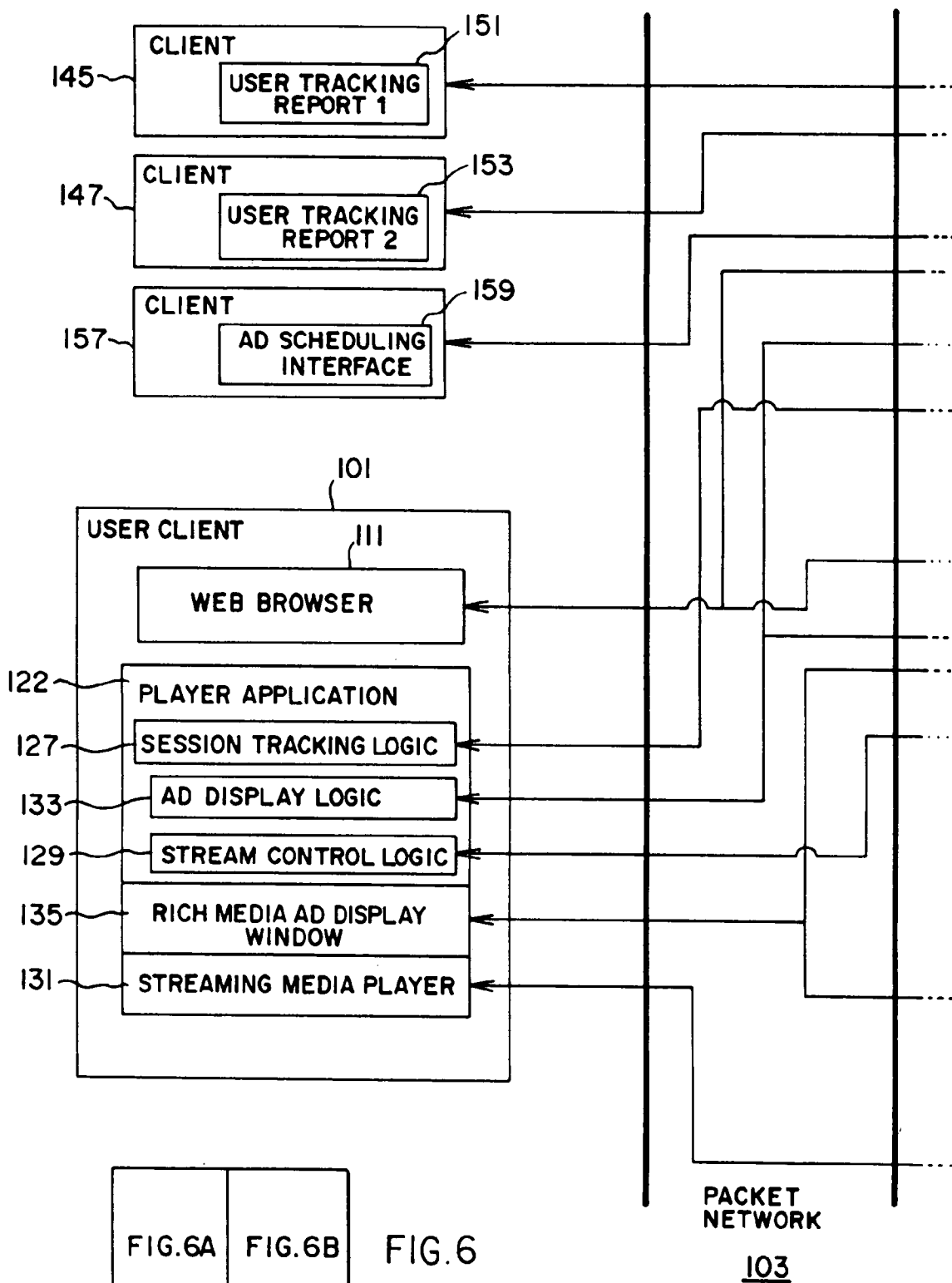


FIGURE 5

FIG. 6A



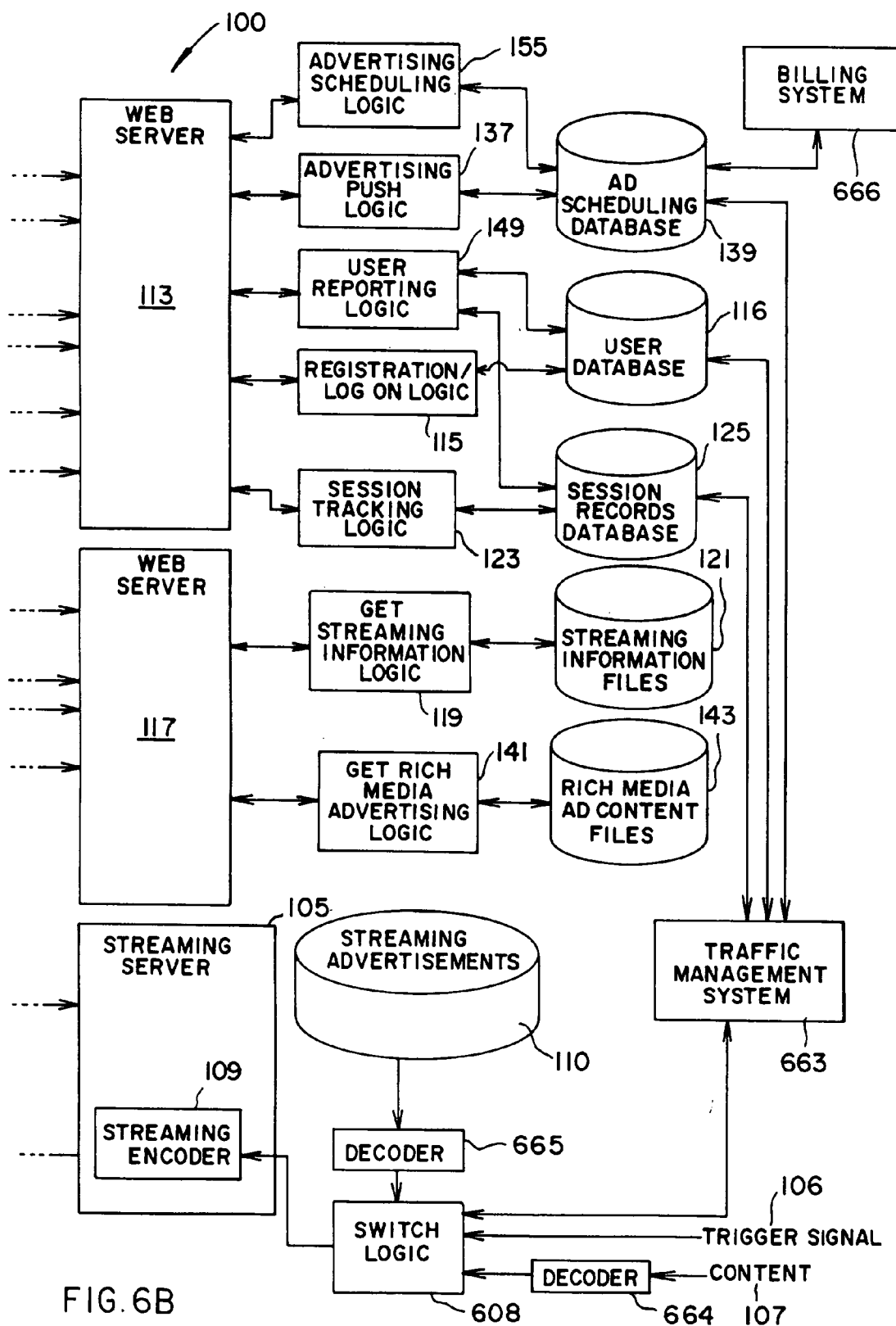


FIG. 6B

METHOD AND APPARATUS FOR SELECTING STREAMING MEDIA IN REAL-TIME

RELATED APPLICATIONS

[0001] The present application is a continuation-in-part of commonly assigned and copending U.S. patent application Ser. No. 09/625,443, entitled "Method and Apparatus for Streaming Media", filed on Jul. 26, 2000, the disclosure of which is hereby incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The invention pertains to streaming media over packet switched data networks, and more particularly to selecting in real-time advertising to be inserted into media streams based on predefined criteria.

BACKGROUND OF THE INVENTION

[0003] Streaming is a process for transmitting audio, video, audio/video and other types of continuous signals, which have been digitized, over packetized data networks such as the Internet for nearly contemporaneous playback. A signal is streamed by encoding the signal as a series of data packets and sending the data packets over a packet switched data network in a manner that supports contemporaneous or nearly contemporaneous playback on a host computer using a player application. Because there are no quality of service or deliver guarantees provided by currently adopted Internet protocols, streaming applications must provide mechanisms for dealing with lost and delayed packets, flow control and encoding and compression, among other problems. Presently, there are several streaming standards and approaches, including those used by the RealPlayer® of RealNetworks, Inc, the Windows Media Player™ of Microsoft Corporation, and the QuickTime® player of Apple Computer, Inc., for encoding and controlling the stream. Pre-recorded content, such as sound recordings and video tapes, and "live" content, such as retransmission of radio and television broadcasts, are presently being transmitted over the Internet using streaming. Graphical advertisements are also transmitted for displaying on a computer screen in connection with the playing of the media stream on the computer. In addition, audio, video or other streaming media advertisements are sometimes transmitted prior to transmission of the content.

SUMMARY OF THE INVENTION

[0004] The invention has as a general objective improved methods and apparatus for a system of streaming audio and/or video signals, and in particular improvements concerning the use of advertising in connection with such streaming.

[0005] According to one feature of an embodiment of a system for streaming audio and/or video signals described below, audio advertisements are inserted into a third party content signal, such as a terrestrial radio broadcast, at a point at which the signal is being turned into a data stream for transmission across, at least in part, a packet switched network, such as the Internet, to a user's computer for contemporaneous playback. The insertion takes place during the streaming, not just at the beginning of the streaming as prior art methods have done. Thus, advertising may be inserted, for example, in place of advertising contained in the original signal. Advertising in a terrestrial radio broad-

cast, which is targeted to a local audience, can be replaced in real time, during streaming, with advertising targeted for a different audience, such as a national audience or an audience with a different demographic profile. To enable insertion or overlaying of advertisements, a trigger signal received from a content provider causes the streaming to switch between a third party content signal and a local signal containing an audio or audio/video insert. In the preferred embodiment, a first trigger signal is received indicating that a second trigger signal will soon be received. In the preferred embodiment, it is the receipt of the second trigger signal that causes the streaming to switch between the third party content signal and the local signal containing an audio or audio/video insert. Furthermore, according to another inventive feature, graphical advertising files can also be transmitted for display on the user's computer, for example in a web browser application and/or streaming media player, in conjunction with the streaming advertisement.

[0006] According to another feature of the embodiment of the system for the streaming audio and/or video signals described below, users of media streams are tracked as the stream is being played, thereby enabling real-time collection of "time-line" information on a stream's audience, including exposure to any advertisements placed in the stream and any non-streaming advertisements displayed on a computer in connection with the media stream. This information may include how many people are listening or viewing a stream at any given time, and how long they have been listening. The invention thus is able to provide information on users that is more accurate than sampling methods like those employed in traditional media. It is also more accurate than tracking only the commencement of a stream or a user "clicking through" a graphical advertisement displayed simultaneously with the stream. Pricing for the advertising inserted into a media stream can thus be determined based on the actual number of users who hear and/or see the advertisements. Real time reporting on users may also be made available to content providers and advertisers. By further obtaining demographic information from a user, the invention may be further used to generate real-time information on the demographic composition of an audience. Such real time demographic information may be used to select in real-time advertising for insertion into the stream or display of graphical advertising at the host computer, or both. Such information may also be used to determine pricing for the advertising. Furthermore, selection of advertising in real-time may additionally or alternatively be based on other predefined criteria such as product code separation, frequency of play of a particular advertisement, the interests of the audience and/or the like.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Following is a detailed description of a method and apparatus for streaming audio and/or video signals, made in reference to the accompanying drawings, of which:

[0008] FIG. 1 is a block schematic diagram of a client server system for streaming audio and video, tracking users and pushing rich media advertising;

[0009] FIG. 2 is a flow diagram representing a set up process for a streaming service provided with the system of FIG. 1;

[0010] FIG. 3 is a flow chart representing a process of a user client for updating advertising displayed in connection with playing of the stream;

[0011] FIG. 4 is a preferred embodiment flow chart representing a process by which a server updates advertising displayed in connection with the playing of the stream;

[0012] FIG. 5 is a flow chart for selecting and playing advertisements in real-time based on predefined criteria; and

[0013] FIG. 6 is a block schematic diagram of a client server system for streaming audio and video, tracking users, selecting advertisements in real time and pushing rich media advertising.

DETAILED DESCRIPTION

[0014] In the following description, like numbers refer to like parts.

[0015] When used herein, the term “computer” refers to any device capable of communicating over a data network and decoding for nearly simultaneous playback of an incoming data stream that is encoded with audio and/or video signals. Such a stream is referred to herein as a media stream. The audio and/or video signals, once decoded, may be played back on the computer or another device for reproducing the sound and/or video represented by the signals. A computer may further include or be associated with a visual display. In the preferred embodiments described herein, a computer takes the form of a microprocessor-based personal computer, that includes a general purpose microprocessor, temporary program and data storage, such as random access memory, permanent program and data storage, such as a disk drive, a monitor or other visual display for displaying graphics, a sound card for decoding and converting digital signals to analog signals, and a keyboard and/or mouse for receiving data from a user. However, computers may also include limited function “Internet appliances having limited display, data, data input, and user programming capabilities, such as personal organizers, telephones and other limited or special purpose devices.

[0016] The term “packet network” refers generally to one or more interconnected public and/or private networks that route packets or frames of data, as opposed to circuit switched networks and television or radio broadcast networks. Packet Network includes the system of interconnected computer networks known as the “Internet” that route data packets using the Internet Protocol (IP) as it exists presently or in the future.

[0017] Referring now to FIG. 1, streaming system 100 provides a streaming service in which it transmits, or causes transmission of, audio, and/or audio/video signals as a data stream. A client computer 101 functions as a device for a user to enjoy the streaming service. Client computer 101 is connected directly or indirectly, such as through a dial up connection, a wireless gateway, a cable modem, a xDSL modem, or local area network, to packet network 103. The data stream is transmitted by a streaming server 105 through packet network 103 to the client computer. Although only one client computer 101 is illustrated for purposes of explanation, the same media stream may be transmitted to a large number of client computers or the server may be transmitting media streams with differing content to different computers.

[0018] The streaming server receives a content signal 107 from a source and transmits the signal as a stream to packet network 103. The signal source may be a terrestrial radio station or television station, or other service that provides audio and/or video programming content. For reasons explained below, the system 100, in its preferred embodiment, may be used to best advantage in transmitting live radio broadcasts. Streaming encoder 109 digitizes, and if desirable, formats and encodes the signal as a stream. Any type of data transport mechanism may be used to transmit the content signal to system 100, including those that transmit the signal in a digital format. Other processes, not represented on the figure, handle the transport of the media stream over the connection of the streaming server to the packet network.

[0019] Content for a media stream, meaning an audio and/or video signal, is in the preferred embodiment provided in real time from a source. In the preferred embodiment, if the source of content signal 107 is a broadcast radio station or radio network, the signal that is broadcast is also being provided in real time for immediate streaming. Once the signal arrives, an audio automation system immediately connects it to the streaming encoder 109. The streaming encoder is in a preferred embodiment, an instance of a server component of any streaming application, such as Real-Player®, QuickTime® or Windows Media Player™. In order to insert advertising into the stream, in real time, a trigger signal 106 from the source is also received. The trigger signal can be sent from the source separately, such as on a different channel, sent on the same channel as or otherwise encoded in the content signal from the source. The trigger signal indicates the start of a time period in which a message, such as an advertisement, a news item, stock alerts, an email message, a weather update, a voice mail message and/or the like may be inserted into the content. For example, the message may be an audio advertisement for a radio signal, or an audio/video advertisement for a television signal.

[0020] In the preferred embodiment, when audio automation system 108 receives a trigger signal, it plays an advertisement that has been queued according to a schedule or a predetermined order. In an alternative embodiment, the advertisements may be selected in real-time based on a set of one or more predefined criteria as discussed in more detail with reference to FIG. 5. The advertisements are stored in storage system 110. The playback of the advertisement is switched by the audio automation system to streaming encoder 109 in place of content signal 107. Thus, it is sent to all users receiving a stream from the URL that identifies the source of the stream. Once the advertisement is finished, audio automation system 108 switches back to the content signal 107 to provide a signal to the streaming encoder 109, or plays additional advertisements. A second trigger signal can be sent to indicate conclusion of the time period for advertisements, or the periods can be set to have a predetermined duration. In the event that the content signal is provided by a third party subscription service, the trigger can be used to signal the start of a new program that may permit insertion of an advertisement. An identifier included in the stream is changed to indicate that a different stream, namely the advertisement, is being sent.

[0021] Although not shown, streaming server 105 may communicate the media stream to other servers and/or one

or more distribution networks that are connected to, or part of, packet network **103**, in order to cache and/or geographically distribute the stream over high speed networks for purposes of enhancing delivery of the signal to each client computer **101**. The stream may also be cached by these other services or networks.

[**0022**] The streaming server **105** may also receive signals from more than one source and concurrently transmit more than one media stream. Furthermore, more than one streaming server may be used to transmit additional media streams.

[**0023**] Referring now to **FIGS. 1 and 2**, to begin use of the streaming service, a client-server application such as the World Wide Web (or “web”) is used to exchange information with the user for setting up the service. The following description will be made in reference to a web server and a web browser as an example of a client-server application used to obtain information about streaming services and to setup streaming services. The Web has an advantage of being available for almost every type of computer. However, other client-server applications can be used to exchange set-up information for streaming services. Therefore, the web browser and web server can be replaced by other types of applications capable of displaying text and/or graphic information, such as those that may be required for computers with limited display or computing capabilities. Client computer **101** therefore includes a web browser **111**. The user obtains a web page, such as an HTML encoded file, on which one or more links to streaming services content are included. At step **201** of the process illustrated by the flow diagram of **FIG. 2**, the user requests a stream by, for example, selecting a hyperlink on a web page. The web browser sends to web server **113** a user identifier, if there is one stored in a special user file on the client computer. Web server **113** passes the information to registration/log on logic **115**, which then validates the user identifier. If no valid user identifier, or no user identifier at all, is sent, the registration process causes, as represented by decision step **203**, the web server **113** to transmit at step **205** a registration page to web browser **111**. At step **207**, the web browser on the client computer displays the registration page. The registration page requests certain information and includes a form into which information is entered. Preferably, it includes information with which to identify the user, such as an Email address, a telephone number, a credit card number, a digital signature and/or other like information. With such identifying information, the opportunity for duplicate registrations can be reduced. Furthermore, the identifying information, such as the Email address can be, if desired, authenticated. The registration page or process may also, if desired, seek from the user certain demographic information, such as age, gender, income, place of residence, ethnicity, languages spoken, interests and/or the like.

[**0024**] In the preferred embodiment in step **209**, the user sends the registration information to web server **113**. Upon receiving the registration information in step **211**, server **113** passes at least a portion of the received information to registration/log on logic **115** to be stored in user record database **116** in step **213**. The database generates a unique user identifier that is sent to the client computer **101** in step **215**.

[**0025**] Once a user identifier is stored on the client computer, the web browser continues with the process at step

217 of setting up the selected media stream for the user. The web browser sends a request to a second web server **117**, using information associated with the link selected by the user at step **201**, for information with which to set up the media stream. Included is information with which to identify the stream. When, as represented by step **219**, this request is received by web server **117**, the stream identifying information is passed to get streaming information logic **119**, which then obtains the appropriate file stored in streaming information file directory or database **121**. This file is transmitted by web server **117** to client computer **101**. At step **221**, the client receives and stores the file. In the preferred embodiment, this file includes a locator, such as a Universal Resource Locator (URL), from which the particular stream is available. Receiving this file causes, in the preferred embodiment, a player application **122** to be launched on client computer **101**.

[**0026**] As represented by steps **223** and **225**, once a user identifier is obtained, session tracking logic **123** creates a record in session records database **125** to track the user's session with the selected media stream. This session record includes, but is not limited to, fields for the user identifier, the time the media stream was set up, and/or information that identifies the media stream (e.g. the radio station broadcast including for example the particular advertisement) sent to the user. A session identifier that uniquely identifies the session is also generated and sent by session tracking logic **123** to client computer **101** in step **227** for storage by the client computer in step **229**. The client computer **101** preferably stores the session identified in client session tracking logic **127**.

[**0027**] Beginning with step **231** the client requests the stream from the URL provided in the file received at step **221**. For purposes of this description, the URL points to a streaming service on streaming server **105**, which is transmitting the stream from source **107**. The streaming server begins transmitting the stream to client computer **101** in step **233**, which preferably includes a stream identifier that is stored by the stream control logic **129** of player application **122** on client computer **101** in step **235**. Player application **122** has embedded in or linked to it a streaming media client **131**, such as Windows Media Player™, that actually controls the streaming and processes and decodes the stream in step **237** for playback on client computer **101** using its sound system and/or a connected sound system. If, as represented by decision steps **239** and **241**, the stream terminates, the stream identifier is deleted at step **245**. If a new stream identifier is received, it is stored in step **243** and the playing process continues at step **237**. A session identifier for a streaming session can be used in place of the stream ID to identify the stream that the user is then currently receiving.

[**0028**] Referring now to **FIGS. 1 and 3** in step **301**, player application **122** sends information to web server **113**. This information may be automatically sent on a periodic basis. This information preferably includes the user identifier, the session identifier and the stream identifier. This information is used by ad display logic **133** as part of a request sent to tracking web server **113** for an updated URL at step **303** for a rich media message, such as an advertisement. This rich media advertisement may include text, static graphic components, and/or active components, and may come from any third party. For example, such components may be for example a video component in MPEG, QT, MOV or other

format, a presentation in Flash, an animated GIF and/or the like. It is preferably displayed in a rich media advertising window **135**, which is in the preferred embodiment a web browser window that is displayed adjacent a window containing controls (such as volume controls) for player application **122** on the monitor of the client computer. Thus, when a user is receiving a media stream, the user is also viewing an advertisement. The ad display logic **133** can be implemented either as a periodic web page refresh or through client/server software. Once the URL for the rich media advertisement is received in step **305**, the rich media advertisement is requested in step **307**. In step **309**, at least a portion of the received rich media advertisement is displayed. The player application waits for a prescribed time before repeating the process, as indicated by step **311**.

[**0029**] The preferred embodiment flow diagram of **FIG. 4** represents the process on tracking server **113** that corresponds to the process represented by the flowchart of **FIG. 3** that takes place on the client computer **101**. This process will be described in connection with web server **113**. However, as previously explained, other client/server software can be used to implement this process. Web server **113** waits, as indicated by decision step **401**, to receive updated information from the computer of each user that uses the streaming services. When it receives updated information, web server **113** passes the information to session tracking logic **123**, for updating the session record for the particular user in step **403**. Advertising push logic **137** also receives information about the stream that is being played. If, as represented by step **405**, the user is playing a message, such as an advertisement, that has been inserted into the stream according to a process that will be described below, the identity of the streaming advertisement is available as a stream or session identifier, as indicated at step **407**. Preferably the stream identifier is used to look up in the advertising scheduling database **139** or some other database the URL of a rich media advertisement that is to be shown at the same time streaming advertisement is played. In step **409**, the advertising scheduling database **139** is updated with this information so that, when the advertising push logic **137** polls the database at step **411** for the URL of the advertisement to be shown, the URL for this rich media advertisement is transmitted to the client at step **413**. If the user was not playing a streaming advertisement at step **405**, but was playing such an advertisement during the last update, then it updates advertising schedule database to transmit the rich media advertisement when the advertising push logic polls the advertising schedule database, as indicated by steps **415** and **417**. In a preferred embodiment the advertising schedule database is updated with a previously scheduled advertisement. However, if desired, in alternative embodiments the advertising schedule database may be updated with an advertisement selected in real-time based on one or more predefined criteria as discussed in more detail with reference to **FIG. 5**. If no streaming advertisement was being played on the prior update, then step **417** is skipped. Web server **117** is illustrated as providing the rich media advertising files. If the URL provided by the process of **FIG. 4** points to a rich media advertising file stored in database or file system **143**, then get rich media ad logic **141** retrieves the files for the advertisement and provides them to web server **117** to send. However, the URL may also point to any other resource on packet network **103**.

[**0030**] Referring now only to **FIG. 1**, client computers **145** and **147**, each running a web browser, are representative of a feature that permits remote viewing in real time, through a public packet network, how many people are actually listening to the content, as well as when they listened and how long they listened. This feature may be made available to the content providers and to advertisers, as it also indicates who has listened to and/or viewed advertisements. Thus, it is possible to charge advertisers based not only on how many people actually saw or heard an advertisement, but also their demographic profile. In the illustrated example, user reporting logic **149**, in response to a request from client computers **145** and **147** to web server **113**, has generated different user tracking reports **151** and **153**. The reports have been sent by web server **113** for display preferably in web browsers on those client computers.

[**0031**] Additionally, advertising can also be scheduled remotely using web server **113** and advertising scheduling logic **155**. The advertising scheduling logic creates an interface **159** that is displayed on client computer **157**. The interface permits adding, modifying and deleting advertising schedules for the rich media advertising.

[**0032**] Web servers **113** and **117** do not necessarily correspond to physical machines. Rather, they represent different instances of a web server, which may or may not be running on the same physical hardware. Similarly, one or more instances of web servers may be used, and multiple instances may be distributed in terms of physical location, depending on loads or other needs of the service provider or particular implementation. The logic that is illustrated—namely advertising scheduling logic **155**, advertising push logic **127**, user reporting logic **149**, registration/log on logic **115**, session tracking logic **123**, get streaming information logic **119**, and get rich media advertising logic **141**—represent classes of computer programs or scripts which may have many instances at any given time. They may or may not run on the same physical hardware as the web servers and thus, too, may be distributed. For example, in the preferred embodiment, that may be instances of dynamic link libraries that are invoked by the web browsers. The lines extending between the various entities in **FIG. 1** indicate message and data flows, and not physical connections. Player application **122** may, alternatively, be implemented as a web page with active components, with the rich media advertisements displayed in a frame.

[**0033**] Referring now to **FIGS. 5 and 6**, an alternative embodiment for a client server system **600** for streaming audio and video, tracking users, selecting advertisements in real time and pushing rich media advertising is shown. The block schematic diagram of **FIG. 6** is substantially the same as the diagram of **FIG. 1**. However, the client server system of **FIG. 6** differs in ways described below. A decoder **664** is provided such that if the content signal is in a coded or compressed format, the decoder **664** decodes or decompresses the signal. Furthermore a decoder **665** is preferably included between a switch logic **608** and streaming advertisements database **110**. Advertisements in the streaming advertisements database **110** are preferably stored in a compressed format and are decoded by decoder **665** prior to providing to switch **608**.

[**0034**] As indicated by decision step **501** traffic management logic **663** waits to receive an indication that a trigger

has been received. Traffic management logic 663 is implemented, for example, as a program or multiple programs running on one or more computers. In step 502 one or more streaming advertisements are selected, based on one or more predefined criteria, to be played during a commercial break in the content. In step 503, the selected advertisement(s) are indicated to or identifies for the switch logic 608. Switch logic 608, like audio automation system 108, detects the trigger signal. However, it also requests from the traffic management system a streaming advertisement to play and then retrieves it from the streaming advertisements database 110. The switch logic can be a programmed process on a computer with multiple sound cards.

[0035] In the preferred embodiment, the traffic management system 663 writes to the ad scheduling database 139 information identifying which streaming advertisement was played and when it was played. Other information, such as the criteria used to select the advertisement and information for determining which rate to be charged to the advertiser, can also be written to the ad scheduling database 139 for use by the billing system 666 to create statements or bills for the advertisers. This information may include the time of day, the number of users who received the streaming advertisement (referred to as "impressions"), the demographic information of the users, the station identifier, the spot number and/or the like. In the preferred embodiment, the cost of the advertisements are calculated using rates based on cost per thousand impressions multiplied by the number of impressions.

[0036] In a preferred embodiment, the demographic composition of the users to whom a particular streaming media is being transmitted is used as a criteria to select the streaming advertisement. The traffic management logic 663 preferably determines the demographic composition of the users listening to a particular stream. Thus, selection of an advertisement for streaming may be, if desired, based on whether the demographic composition of the users matches or fits the demographic profile associated with the particular advertisement. In the illustrated embodiment the demographic composition of the users is determined by the traffic management logic 663. In the illustrated embodiment, the traffic management logic 663 accesses the session records database 125 and acquires the user identifiers of a plurality of users associated with a particular streaming session. The user identifiers are used by the traffic management logic 663 to look up the profile of the users stored in the user records database 116.

[0037] Information about the preferred target audience of a particular advertisement may be associated with the advertisement and stored in the ad scheduling database 139 along with the particular advertisement. Thus, information from the user database 116 may be used to select a particular advertisement to be played. Thus, whether a particular advertisement is selected for playing will depend, at least in part, on the demographic profile, such as for example the age, gender, income, place of residence, ethnicity, interests and/or the like of the users. For example, an advertiser may have two "spots", one targeted for one demographic and the other targeted to a different demographic. The most appropriate advertisement can be selected based on which demographic is most prevalent among the users. Furthermore, if desired, each of these demographic criteria may be given a particular weight in the selection process such that a par-

ticular demographic criteria is given more importance in the selection process. For example, advertisers for local goods and/or services may be more interested in the geographical location of a user, then their income and thus, may assign a greater weight to the place of residence.

[0038] In the illustrate embodiment, predefined criteria may be used to enforce product code separation. A product code indicates the product that a particular streaming advertisement is related to. For example, the streaming advertisement database may include one or more advertisements for cars from different manufacturers. The product code for all such advertisements may be the same indicating that all of the advertisements relate to cars irrespective of the manufacturer. A product code separation criteria can be used to enforce an advertiser's requirement or preference that some number of advertisements or some amount of time pass between playing of advertisements of the same product class or type. Ad scheduling database 139 may include information associated with each advertisement, such as a particular advertiser's preferences as to product code separation. In such a case, a particular advertiser could specify for example the number of advertisements or a time period that would separate that particular advertiser's advertisement from an advertisement related to a product having the same or similar product code. Use of a product code separation will, when selecting advertisements for play in real-time, prevent inadvertent violation of the product separation requirements or preferences.

[0039] Additional predefined criteria for selecting may be based, for example on the frequency of play. Thus, whether a particular advertisement is selected for playing in real time could depend on when that particular advertisement was last played. The information regarding when a particular advertisement was last played could be obtained for example from the session records database 125 and/or the session tracking logic 127. In the preferred embodiment, the same advertisement is not played around the same time everyday. Thus, if a particular advertisement was played at a particular time the previous day, then in the preferred embodiment, that same advertisement would not be played during the same time the next day.

[0040] Alternately, the selection of the advertisement may precede actual receipt of a trigger to improve performance, or for other reasons, provided the selection is made in close proximity to receiving the trigger signal. The advertisement could be streamed to the users upon receiving the trigger. For example, because in the preferred embodiment, user session records are only updated periodically, the traffic management logic 663 need only periodically generate the demographic composition of the users. This could improve the performance of the system in providing advertisements in real time. Furthermore, if desired, some of the predefined criteria may be calculated or applied prior to receiving the trigger signal while other predefined criteria may be applied after receiving the trigger signal.

[0041] In a preferred embodiment, during the selection process a first criteria, for example the demographic composition of the users is applied to a plurality of advertisements, say advertisements scheduled to be played during a particular period of the day (for example, evening drive time) to provide a subset of the plurality of advertisements. A second criteria, say frequency of play, may then be applied

to this subset of advertisements to provide a smaller subset of advertisements. A third criteria, say product code separation, may then be applied to this smaller subset to select one or more advertisements to be played. If there are more than one advertisements that meet all three criteria, then these advertisements could be played based on a priority basis or some other bases.

[0042] Although the real time selection of streaming advertisements is described above with reference to certain criteria only, any one or any number of criteria, as well as other criteria, may be used to select a streaming advertisement in real time. Moreover, it is not necessary that all the criteria be considered in selecting the particular advertisement to be displayed. Any one or more than one combination of criteria may be used for selecting an advertisement to be played. Furthermore, weights may be assigned to each of the predefined criteria so that a particular predefined criteria is given more importance in the selection process than other criteria.

[0043] In alternative embodiments, a set of advertisements may be queued in advance with alternate "spots" provided based on predefined criteria such as demographics, or with the predefined criteria acting as a screen to prevent playing of the advertisement. For example, advertisement numbers 1, 2, 3, 4 and 5 may be scheduled to be delivered in that order in a particular time slot. However, if the traffic management system determines that advertisement 2 is targeted to teenage girls while the particular listeners are retired males, advertisement 2 can be skipped and advertisement 3 is played or an alternate spot played in place of advertisement 2.

[0044] The advertisement selection process as described above with respect to the flowchart of FIG. 5 provides certain advantages not provided by prior art systems. For example, because it is capable of selecting in real-time advertisements to be played, the advertisements may be better targeted to an advertiser's preferred audience. Furthermore, because a user is receiving advertisements for products and/or services that it is interested in, the user is able to receive better information. This may provide a competitive advantage to the particular content provider, such as a radio station, because the users would prefer receiving content from a content provider that also provides them useful information during commercial breaks than a content provider who does not provide them useful information during commercial breaks.

[0045] The forgoing description is an example of one embodiment of the invention. The invention is not, however, limited to the described and illustrated embodiment. Elements and features of this embodiment may be omitted or altered, and features and elements added, without departing from the scope of the invention, which is defined solely by the appended claims.

What is claimed is:

1. A method for streaming media over packet networks, comprising the steps of:

receiving a first content signal from a source;

establishing a streaming session with a user computer over a packet switched data network;

streaming said first content signal over said data network during said streaming session;

in response to a trigger, selecting in real time a second content signal to be streamed in place of said first content signal; and

streaming said selected second content signal.

2. The method of claim 1, wherein said second content signal includes advertising information.

3. The method of claim 2, wherein said selection in real time of said second content signal is based in part on a set of predefined criteria.

4. The method of claim 3, wherein at least one criteria of said set of predefined criteria is selected from the group consisting of a product code separation, a frequency of streaming, and a demographic profile of a user of said user computer.

5. The method of claim 4, wherein said product code separation is based in part on a stream identifier associated with said streaming session, wherein said stream identifier identifies a particular product associated with said advertising information.

6. The method of claim 3, wherein said advertising information is selected from a plurality of advertisements stored in a streaming advertisements database, wherein said plurality of advertisements are prearranged in a scheduled order and said selecting step rearranges said prearranged order based on said set of predefined criteria.

7. The method of claim 2, wherein said advertising information includes an audio component.

8. The method of claim 2, wherein said advertising information includes a video component.

9. The method of claim 7, wherein said audio component is in MP3 format

10. The method of claim 8, wherein video component is in a format selected from the group consisting of MPEG, MOV, QT, and animated GIF.

11. A computer program for providing streaming media over a packet switched network to a user computer, the computer program comprising:

a user component for use by a user computer; and

a provider component on a provider server for use by a provider of said streaming media, wherein said provider component includes:

code for receiving relevant information from said user component;

code for streaming a content signal received from a source to said user component;

code for receiving a trigger signal from said source;

code for selecting, upon receiving said trigger signal, in real time content information to be provided to said user computer based in part on information received from said user component over a packet switched data network, wherein said selection is based on a predefined set of criteria; and

code for streaming said selected content information over said data network to said user computer.

12. The computer program of claim 11, wherein said selected information includes a plurality of advertisements and wherein said provider component further comprises:

code for rearranging a prearranged order of streaming said plurality of advertisements, wherein said rearranging is based on said predefined set of criteria.

13. A system for streaming media over packet networks, comprising:

means for receiving a first content signal from a source;

means for establishing a streaming session with a user computer over a packet switched data network;

means for streaming said first content signal over said data network during said streaming session;

means for streaming, in response to a trigger, a second content signal, wherein said second content signal to be streamed is selected in real time.

* * * * *

APPENDIX B-5

[MS-WMLOG]: Windows Media Log Data Structure

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

| Date | Revision History | Revision Class | Comments |
|------------|------------------|----------------|--|
| 04/03/2007 | 0.01 | | MCPP Milestone Longhorn Initial Availability |
| 07/03/2007 | 1.0 | Major | MLonghorn+90 |
| 07/20/2007 | 2.0 | Major | Revised technical content; added example topics. |
| 08/10/2007 | 2.0.1 | Editorial | Revised and edited the technical content. |
| 09/28/2007 | 2.0.2 | Editorial | Revised and edited the technical content. |

| Date | Revision History | Revision Class | Comments |
|-------------|-------------------------|-----------------------|---|
| 10/23/2007 | 2.0.3 | Editorial | Revised and edited the technical content. |
| 11/30/2007 | 2.0.4 | Editorial | Revised and edited the technical content. |
| 01/25/2008 | 2.0.5 | Editorial | Revised and edited the technical content. |
| 03/14/2008 | 2.1 | Minor | Updated the technical content. |
| 05/16/2008 | 2.1.1 | Editorial | Revised and edited the technical content. |
| 06/20/2008 | 2.2 | Minor | Updated the technical content. |

Table of Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 5 |
| 1.1 | Glossary | 5 |
| 1.2 | References | 5 |
| 1.2.1 | Normative References | 5 |
| 1.2.2 | Informative References | 6 |
| 1.3 | Relationship to Protocols and Other Structures | 6 |
| 1.4 | Applicability Statement | 7 |
| 1.5 | Versioning and Localization | 7 |
| 1.6 | Vendor-Extensible Fields | 7 |
| 2 | Structures..... | 8 |
| 2.1 | Log Data Fields | 8 |
| 2.1.1 | audiocodec | 8 |
| 2.1.2 | avgbandwidth | 9 |
| 2.1.3 | c-buffercount | 9 |
| 2.1.4 | c-cpu | 9 |
| 2.1.5 | c-dns | 10 |
| 2.1.6 | c-hostexe | 10 |
| 2.1.7 | c-hostexever..... | 10 |
| 2.1.8 | c-ip | 10 |
| 2.1.9 | c-max-bandwidth | 11 |
| 2.1.10 | c-os | 11 |
| 2.1.11 | c-osversion | 11 |
| 2.1.12 | c-pkts-lost-client | 12 |
| 2.1.13 | c-pkts-lost-cont-net | 12 |
| 2.1.14 | c-pkts-lost-net | 13 |
| 2.1.15 | c-pkts-received | 13 |
| 2.1.16 | c-pkts-recovered-ECC | 13 |
| 2.1.17 | c-pkts-recovered-resent | 14 |
| 2.1.18 | c-playerid | 14 |
| 2.1.19 | c-playerlanguage..... | 15 |
| 2.1.20 | c-playerversion | 15 |
| 2.1.21 | c-quality..... | 15 |
| 2.1.22 | c-rate..... | 16 |
| 2.1.23 | c-resendreqs..... | 16 |
| 2.1.24 | c-starttime | 17 |
| 2.1.25 | c-status..... | 17 |
| 2.1.25.1 | Status Code 200 (No Error) | 17 |
| 2.1.25.2 | Status Code 210 (Client Successfully Reconnected) | 17 |
| 2.1.26 | c-totalbuffertime | 17 |
| 2.1.27 | c-channelURL..... | 18 |
| 2.1.28 | c-bytes..... | 18 |
| 2.1.29 | cs-media-name | 18 |
| 2.1.30 | cs-media-role..... | 19 |
| 2.1.31 | cs-Referer | 20 |
| 2.1.32 | cs-url | 20 |
| 2.1.33 | cs-uri-stem..... | 20 |
| 2.1.34 | cs-User-Agent | 21 |
| 2.1.35 | cs-user-name..... | 22 |
| 2.1.36 | date..... | 22 |
| 2.1.37 | filelength | 22 |
| 2.1.38 | filesize | 22 |

| | | |
|----------|--|-----------|
| 2.1.39 | protocol..... | 23 |
| 2.1.40 | s-content-path | 23 |
| 2.1.41 | s-cpu-util..... | 23 |
| 2.1.42 | s-dns | 24 |
| 2.1.43 | s-ip | 24 |
| 2.1.44 | s-pkts-sent..... | 25 |
| 2.1.45 | s-proxied..... | 25 |
| 2.1.46 | s-session-id | 25 |
| 2.1.47 | s-totalclients..... | 25 |
| 2.1.48 | sc-bytes | 26 |
| 2.1.49 | time..... | 26 |
| 2.1.50 | transport..... | 26 |
| 2.1.51 | videocodec | 27 |
| 2.1.52 | x-duration | 27 |
| 2.2 | Logging Message: W3C Syntax | 28 |
| 2.2.1 | Basic Logging Syntax | 28 |
| 2.2.2 | Extended Logging Syntax | 28 |
| 2.2.3 | Connect-Time Logging Syntax..... | 29 |
| 2.3 | Logging Messages Sent to Web Servers | 29 |
| 2.4 | Logging Message: XML Schema | 30 |
| 2.5 | Legacy Log..... | 32 |
| 2.5.1 | Common Definitions..... | 32 |
| 2.5.2 | Legacy Log in W3C Format | 34 |
| 2.5.3 | Legacy Log in XML Format | 34 |
| 2.5.4 | Legacy Log Sent to a Web Server | 34 |
| 2.6 | Streaming Log..... | 34 |
| 2.6.1 | Common Definitions..... | 35 |
| 2.6.2 | Streaming Log Sent to Windows Media Services | 36 |
| 2.6.3 | Streaming Log Sent to a Web Server | 36 |
| 2.7 | Rendering Log | 37 |
| 2.7.1 | Common Definitions..... | 37 |
| 2.7.2 | Rendering Log Sent to Windows Media Services | 38 |
| 2.7.3 | Rendering Log Sent to a Web Server..... | 38 |
| 2.8 | Connect-Time Log | 39 |
| 3 | Structure Examples | 40 |
| 3.1 | Legacy Logging Message..... | 40 |
| 3.2 | Defining Custom Namespaces in an XML Log | 41 |
| 3.3 | Example Streaming Log Messages..... | 42 |
| 3.4 | Example Rendering Log Messages | 44 |
| 3.5 | Example Connect-Time Log Message | 45 |
| 3.6 | Example Log Sent to a Web Server | 45 |
| 3.7 | Parsing Windows Media Log Files..... | 46 |
| 4 | Security Considerations | 47 |
| 5 | Appendix A: Windows Behavior | 48 |
| 6 | Index..... | 49 |

1 Introduction

This specification defines the Windows Media Log Data Structure, a Microsoft proprietary interface. The Windows Media Log Data Structure is a syntax for logging messages. The logging messages specify information about how a client received multimedia content from a streaming server. For example, logging messages can specify how many packets were received and how long it took for the client to receive the content.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

Advanced Systems Format (ASF)
Globally Unique Identifier (GUID)

The following terms are defined in [\[MS-WMSP\]](#):

Content
Playlist
Session
Stream
Streaming

The following terms are specific to this document:

Client: The entity that has created the logging message, or an entity that receives a logging message from a client. In the latter case, the client is a proxy.

Proxy: An entity that can receive logging messages from both a client and a proxy, and/or a server that is streaming on behalf of another server.

Server: An entity that transfers content to a client through streaming. A server might be able to do streaming on behalf of another server, thus a server can also be a proxy.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[ASF] Microsoft Corporation, "Advanced Systems Format Specification", December 2004, http://download.microsoft.com/download/7/9/0/790fecaa-f64a-4a5e-a430-0bccdab3f1b4/ASF_Specification.doc

If you have any trouble finding [ASF], please check [here](#).

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, <http://www.ietf.org/rfc/rfc1945.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2326] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998, <http://www.ietf.org/rfc/rfc2326.txt>

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>

[RFC3066] Alvestrand, H., "Tags for the Identification of Language", RFC 3066, January 2001, <http://www.ietf.org/rfc/rfc3066.txt>

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", RFC 3629, November 2003, <http://www.ietf.org/rfc/rfc3629.txt>

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

[RFC4234] Crocker, D., Ed. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, <http://www.ietf.org/rfc/rfc4234.txt>

1.2.2 Informative References

[MS-MMSP] Microsoft Corporation, "[Microsoft Media Server \(MMS\) Protocol Specification](#)", June 2007.

[MS-MSB] Microsoft Corporation, "[Media Stream Broadcast \(MSB\) Protocol Specification](#)", January 2007.

[MS-RTSP] Microsoft Corporation, "[Real-Time Streaming Protocol \(RTSP\) Windows Media Extensions](#)", July 2007.

[MS-WMSP] Microsoft Corporation, "[Windows Media HTTP Streaming Protocol Specification](#)", March 2007.

[MSDN-WMMETA] Microsoft Corporation, "Windows Media Metafiles", <http://msdn2.microsoft.com/en-us/library/bb248407.aspx>

[MSFT-LOGPARSER] Microsoft Corporation, "Log Parser 2.2", <http://www.microsoft.com/downloads/details.aspx?FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07&displaylang=en>

[W3C-EXLOG] World Wide Web Consortium, "Extended Log File Format", <http://www.w3.org/TR/WD-logfile.html>

1.3 Relationship to Protocols and Other Structures

The logging messages defined in this specification are used by the [Windows Media HTTP Streaming Protocol](#) and the [Real-Time Streaming Protocol \(RTSP\) Windows Media Extensions](#). When those two protocols are used, the logging messages defined by this specification can be encapsulated in protocol messages specific to the **streaming** protocol in use. The resulting protocol messages are sent to either Windows Media Services or to a proxy compatible with the logging message syntax defined in this specification.

It is also possible to send logging messages to an HTTP Web server. This is possible when using the two streaming protocols mentioned above and when using two other streaming protocols: [Microsoft Media Server \(MMS\) Protocol](#) and [Media Stream Broadcast \(MSB\) Protocol](#).

1.4 Applicability Statement

The syntax for logging messages defined by this specification is applicable to implementations of the four streaming protocols mentioned in section [1.3](#).

1.5 Versioning and Localization

None.

1.6 Vendor-Extensible Fields

Logging messages in XML format are vendor-extensible. Any logging information added by a vendor MUST be encoded using the "client-logging-data" syntax element specified in section [2.4](#).

2 Structures

Section [2.1](#) defines fields that can appear in a logging message. Not all fields appear in all logging messages, however. Section [2.2](#) defines the syntax of W3C-based logging messages, and section [2.4](#) defines the syntax of XML-based logging messages.

Section [2.5](#) defines the legacy logging message type. Section [2.6](#) defines the streaming log message type. Section [2.7](#) defines the rendering log message type. Section [2.8](#) defines the connection log message type.

The information contained in a logging message is always specific to a particular session. The extent of a session is defined by the streaming protocol used by the server. A rendering log message (as specified in section [2.7](#)) can be sent without streaming from a server, and, in that case, a session starts when the playback of the playlist starts, and stops when the playback of the playlist stops.

Below are some common Augmented Backus-Naur Form (ABNF) constructions, as specified in [\[RFC4234\]](#), that are used throughout this specification. Any ABNF syntax rules that are not defined in [\[RFC4234\]](#) or in this specification may be defined in [\[RFC1945\]](#) or [\[RFC2616\]](#).

```
date-year    = 4DIGIT          ; "19xx" and "20xx" typical
date-month   = 2DIGIT          ; 01 through 12
date-day     = 2DIGIT          ; 01 through 31
time-hour    = 2DIGIT          ; 00 through 24
time-min     = 2DIGIT          ; 00 through 59
time-sec     = 2DIGIT          ; 00 through 59, 60 if leap second

ip_addr      = IPv4address | IPv6address
              ; Defined in Appendix A of RFC3986

ver_major    = 1*2DIGIT
ver_minor    = 1*2DIGIT [ "." 1*4DIGIT "." 1*4DIGIT ]
```

2.1 Log Data Fields

2.1.1 audiocodec

This field SHOULD specify a list of audio codecs used to decode the audio **streams** accessed by the **client**. Each codec MUST be listed only once regardless of the number of streams decoded by that codec.

The value for **audiocodec** MUST NOT exceed 256 characters in total length.

The syntax of the **audiocodec** field is defined as follows:

```
codec-name= *VCHAR
audiocodec= "-" | ( codec-name *( ";" codec-name ) )
```

Example:

```
Microsoft_Audio_Codec;Generic_MP3_Codec
```

2.1.2 avgbandwidth

This field MUST specify the average bandwidth, in bits per second, at which the client received **content** from the **server** (which may be a **proxy**), as measured by the client from the start of the current session. This is only applicable during periods in which the server is streaming the content. Depending on the streaming protocol used, it might be possible for the session to be in a "paused" state in which streaming is suspended. The value for **avgbandwidth** does not account for the average bandwidth during such periods in which streaming is suspended.

If the notion of an average bandwidth is not applicable, for example, because the client did not receive any content from the server, then the field MUST be set to "-".

If the numerical value is specified, it MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **avgbandwidth** field is defined as follows:

```
avgbandwidth= "-" | 1*10DIGIT
```

Example:

```
102585
```

2.1.3 c-buffercount

This field MUST specify the number of times the client buffered while playing the content, counted from when the client most recently started streaming the content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-buffercount** field is defined as follows:

```
c-buffercount= 1*10DIGIT
```

Example:

```
1
```

2.1.4 c-cpu

This field MUST specify the type of CPU used by the client computer.

The syntax of the **c-cpu** field is defined as follows:

```
c-cpu= 1*64VCHAR
```

Example:

```
Pentium
```

2.1.5 c-dns

This field SHOULD be set to "-". The field MAY specify the DNS name of the client sending the log. <1>

The syntax of the **c-dns** field is defined as follows:

```
c-dns= "-"
      | reg-name ; as defined in [RFC3986]
```

Example:

```
wmt.test.com
```

2.1.6 c-hostexe

This field specifies the file name of the host application executed on the client. This field MUST NOT refer to a .dll, .ocx, or other non-executable file.

The syntax of the **c-hostexe** field is defined as follows:

```
c-hostexe= *255VCHAR
```

Example:

```
wmplayer.exe
```

2.1.7 c-hostexever

This field MUST specify the version number of the host application running on the client.

The syntax of the **c-hostexever** field is defined as follows:

```
c-hostexever= ver_major "." ver_minor
```

Example:

```
6.2.5.323
```

2.1.8 c-ip

When a client creates a logging message, it SHOULD specify the **c-ip** field as "-" but MAY specify the IP address of the client.

If a proxy is forwarding a logging message on behalf of a client, the **c-ip** field MUST specify the IP address of the client. The proxy MUST replace the value of the **c-ip** field that was specified by the client with the IP address of the client (as known to the proxy).

The syntax of the **c-ip** field is defined as follows:

```
c-ip = "-" | ip_addr
```

Example:

```
157.100.200.300
```

Example:

```
3ffe:2900:d005:f28b:0000:5efe:157.55.145.142
```

2.1.9 c-max-bandwidth

This field MUST be set to "-".

The syntax of the **c-max-bandwidth** field is defined as follows:

```
c-max-bandwidth = "-"
```

Example:

```
-
```

2.1.10 c-os

This field MUST specify the client's operating system. <2>

The syntax of the **c-os** field is defined as follows:

```
OSname= "Windows_98" | "Windows_ME" | "Windows_NT"  
        | "Windows_2000" | "Windows_XP" | "Windows"  
        | "Windows_Server 2003"  
c-os = OSname | 1*64VCHAR
```

Example:

```
Windows
```

2.1.11 c-osversion

This field MUST specify the version number of the client's operating system.

The syntax of the **c-osversion** field is defined as follows:

```
c-osversion= ver_major "." ver_minor
```

Example:

```
6.0.0.6000
```

2.1.12 c-pkts-lost-client

This field MUST specify the number of ASF data packets ([\[ASF\]](#) section 5.2) lost during transmission from server to client and not recovered at the client layer through error correction or at the network layer by using the User Datagram Protocol (UDP) resends, counted from when the client most recently started streaming the content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-client** field is defined as follows:

```
c-pkts-lost-client= 1*10DIGIT
```

Example:

```
5
```

2.1.13 c-pkts-lost-cont-net

This field MUST specify the largest number of ASF data packets that were lost as a consecutive span during transmission from server to client and counted from when the client most recently started streaming the content.

For example, if data packets numbered 1, 4, and 8 are received, and packets 2, 3, 5, 6 and 7 are lost, then packets 2 and 3 constitute a span of two lost packets, and packets 5, 6 and 7 constitute a span of three lost packets. In this example, the **c-pkts-lost-cont-net** field would be set to 3—the size of the largest span.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-cont-net** field is defined as follows:

```
c-pkts-lost-cont-net= 1*10DIGIT
```

Example:

```
2
```

2.1.14 c-pkts-lost-net

This field MUST specify the number of ASF data packets lost on the network layer, counted from when the client most recently started streaming the content. Packets lost at the network layer can be recovered if the client re-creates them by using forward error correction.

The numerical difference between the value of **c-pkts-lost-net** and the value of [c-pkts-lost-client](#) MUST be equal to the value of [c-pkts-recovered-ECC](#).

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-net** field is defined as follows:

```
c-pkts-lost-net= 1*10DIGIT
```

Example:

```
2
```

2.1.15 c-pkts-received

This field MUST specify the number of ASF data packets that have been correctly received by the client on the first attempt counted from when the client most recently started streaming the content. (ASF data packets that were received through error correction code (ECC) recovery or UDP resends are not included in the **c-pkts-received** field.)

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-received** field is defined as follows:

```
c-pkts-received= 1*10DIGIT
```

Example:

```
523
```

2.1.16 c-pkts-recovered-ECC

This field MUST specify the number of ASF data packets that were lost at the network layer but were subsequently recovered, counted from when the client most recently started streaming the content.

The value of this field MUST be equal to the numerical difference between the value of [c-pkts-lost-net](#) and the value of [c-pkts-lost-client](#).

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-recovered-ECC** field is defined as follows:

```
c-pkts-recovered-ECC= 1*10DIGIT
```

Example:

2.1.17 c-pkts-recovered-resent

This field **MUST** specify the number of ASF data packets that were either recovered because they were resent through UDP or because they were received out of order.

The value **MUST** be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-recovered-resent** field is defined as follows:

```
c-pkts-recovered-resent= 1*10DIGIT
```

Example:

```
5
```

2.1.18 c-playerid

This field specifies a unique identifier for the client application that originated the request. The identifier **MUST** be a **GUID**. The GUID is expressed in registry format and is not enclosed in quotation marks, as shown by the ABNF syntax below.

If the client is configured to remain anonymous (that is, not send private information), the client **MUST** set the **c-playerid** field as indicated by the ABNF syntax for the `playid_priv` syntax element as shown in the code example below. Otherwise, **c-playerid** **MUST** use the syntax for `playid_pub` as shown in the code example below. The client **MUST** choose a value for `playid_pub` randomly, and the same value **MUST** be used for `playid_pub` in all logging messages created by the client application, regardless of which content is streamed.

Furthermore, multiple instances, or incarnations, of the client application **MUST** use the same value for the `playid_pub` syntax element. However, if the client application is shared by multiple users, and it is possible to determine a user identity or account name of the user launching the client application, then the value for `playid_pub` **SHOULD** be different for each user identity or account name. For example, multi-user operating systems typically have separate accounts with a distinct account name for each user, while cellular telephones do not.

If the client uses the `playid_priv` syntax element, then the client **SHOULD** choose the value for the `playid` syntax element randomly; however, the client **MUST** use the same `playid` value for all logging messages sent for the same session.

The syntax of the **c-playerid** field is defined as follows:

```
playid= 12HEXDIG
playid_pub = "{" 8HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-"
              4HEXDIG "-" 12HEXDIG "}"
playid_priv= "{3300AD50-2C39-46c0-AE0A-" playid "}"

c-playerid= playid_pub / playid_priv
```

Example:

```
{c579d042-cecc-11d1-bb31-00a0c9603954}
```

Example (client is anonymous):

```
{3300AD50-2C39-46c0-AE0A-70b64f321a80}
```

2.1.19 c-playerlanguage

This field MUST specify the language-country code of the client.

The syntax of the **c-playerlanguage** field is defined as follows:

```
c-playerlanguage= Language-Tag  
; see section 2.1 of [RFC3066]
```

Example:

```
en-US
```

2.1.20 c-playerversion

This field MUST specify the version number of the client.

The syntax of the **c-playerversion** field is defined as follows:

```
c-playerversion = ver_major "." ver_minor
```

Example:

```
7.0.1024
```

2.1.21 c-quality

This field MUST specify the percentage of packets that were received by the client, counted from when the client most recently started streaming the content.

If **cPacketsRendered** represents all packets received by the client including packets recovered by ECC and UDP resend such that:

```
cPacketsRendered = c-pkts-received + c-pkts-recovered-ECC + c-pkts-  
recovered-resent
```

then the value for the **c-quality** field MUST be calculated as:

$[cPacketsRendered / (cPacketsRendered + c-pkts-lost-client)] * 100$

If the denominator in the above equation evaluates to 0, **c-quality** MUST be specified as 100.

The syntax of the **c-quality** field is defined as follows:

`c-quality = 1*2DIGIT / "100"`

Example:

89

2.1.22 c-rate

This field MUST specify the rate of streaming or playback as a multiplier of the normal streaming or playback rate.

For example, a value of 1 specifies streaming or playback at the normal rate, while a value of -5 indicates rewind at a speed 5 times faster than real-time, and a value of 5 indicates fast-forward at a rate 5 times faster than real-time.

For Legacy and Streaming Logs, **c-rate** MUST be the streaming rate. For Rendering logs, **c-rate** MUST be the rendering (playback) rate.

The value of **c-rate** MUST reflect the rate that was in effect at the beginning of the period covered by the logging message because streaming or playback might already have ended by the time the logging message is generated.

The syntax of the **c-rate** field is defined as follows:

`c-rate= ["-"] 1*2DIGIT`

Example:

1

2.1.23 c-resendreqs

This field MUST specify the number of requests made by the client to receive lost ASF data packets, counted from when the client most recently started streaming the content. If the client is not using UDP resend, the value of this field MUST be "-".

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-resendreqs** field is defined as follows:

`c-resendreqs= "-" / 1*10DIGIT`

Example:

5

2.1.24 c-starttime

This field MUST specify the time offset, in seconds, in the content from which the client started to render content. This represents the presentation time of the ASF data packets that the client began rendering. For live broadcasts, the client MUST set this field to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-starttime** field is defined as follows:

```
c-starttime= 1*10DIGIT
```

Example:

39

2.1.25 c-status

This field MUST specify a numerical code that indicates the status of the client that creates the logging message.

The syntax of the **c-status** field is defined as follows:

```
c-status= "200" / "210"
```

Example:

200

2.1.25.1 Status Code 200 (No Error)

This code indicates that the client successfully streamed and submitted the log.

2.1.25.2 Status Code 210 (Client Successfully Reconnected)

This code indicates that the client disconnected and then reconnected to the server. [<3>](#)

2.1.26 c-totalbuffertime

This field MUST specify the total time, in seconds, that the client spent buffering the ASF data packets in the content, counted from when the client most recently started streaming the content. If the client buffers the content more than once before a log is generated, **c-totalbuffertime** MUST be equal to the total amount of time that the client spent buffering the ASF data packets.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-totalbuffertime** field is defined as follows:

```
c-totalbuffertime= 1*10DIGIT
```

Example:

```
20
```

2.1.27 c-channelURL

This field MUST specify the URL to the multicast station (.nsc) file (for more information, see [\[MS-MSB\]](#)) if such a file was used by the client. Whenever an .nsc file is used, this field MUST be specified, even if the MSB Protocol was not used to stream content.

The syntax of the **c-channelURL** field is defined as follows:

```
c-channelURL = "-"  
              / URI-reference ; as defined in section 4.1 of [RFC3986].
```

Example:

```
http://server/channel.nsc
```

2.1.28 c-bytes

This field MUST specify the number of bytes received by the client from the server, counted from when the client most recently started streaming the content.

The value for the **c-bytes** field MUST NOT include TCP/IP or other overhead added by the network stack. Higher-level protocols such as HTTP [\[RFC2616\]](#), RTSP [\[RFC2326\]](#), and the MMS Protocol [\[MS-MMSP\]](#), can each introduce differing amounts of overhead, resulting in different values for the same content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-bytes** field is defined as follows:

```
c-bytes= 1*10DIGIT
```

Example:

```
28000
```

2.1.29 cs-media-name

The purpose of this field is to specify the file name of the content or server-side **playlist** entry that was streamed or played by the client. For Legacy and Streaming Logs, the value of this field MUST

be the content or server-side playlist entry that was streamed. For Rendering Logs, it MUST be the content or server-side playlist entry that was rendered (played).

If the server provided a Content Description, (see, for example, the [Windows Media HTTP Streaming Protocol](#)), and the Content Description contains an entry named WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL, the value of the **cs-media-name** field MUST be equal to the value of the WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL entry.

Otherwise, if the client is using an Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)), and the file specifies a logging parameter called "cs-media-name", then the value of the **cs-media-name** field in the logging message MUST be equal to the value of the "cs-media-name" logging parameter in the .asx file. See section [3.2](#) for an example of how this parameter is specified in an .asx file.

If none of the above applies, **cs-media-name** MUST be specified as "-".

The syntax of the **cs-media-name** field is defined as follows:

```
cs-media-name= *VCHAR
```

Examples:

```
C:\wmpub\wmroot\MyAd2.asf
```

2.1.30 cs-media-role

The purpose of this field is to specify a value that can be associated with a server-side playlist entry to signify the role of the playlist entry. For Legacy and Streaming logs, the value of this field MUST be the role of the server-side playlist entry that was streamed. For Rendering Logs, it MUST be the role of the server-side playlist entry that was rendered (played).

If the server provided a Content Description, (see, for example, the [Windows Media HTTP Streaming Protocol](#)), and the Content Description contains an entry named WMS_CONTENT_DESCRIPTION_ROLE, the value of the **cs-media-role** field MUST be equal to the value of the WMS_CONTENT_DESCRIPTION_ROLE entry.

Otherwise, if the client is using an Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)), and the file specifies a logging parameter called "cs-media-role", then the value of the **cs-media-role** field in the logging message MUST be equal to the value of the "cs-media-role" logging parameter in the .asx file. See section [3.2](#) for an example of how this parameter is specified in an .asx file.

If none of the above applies, the **cs-media-role** MUST be specified as "-".

The syntax of the **cs-media-role** field is defined as follows:

```
cs-media-role= *VCHAR
```

Example:

```
ADVERTISEMENT
```

2.1.31 cs-Referer

This field SHOULD specify the URL to the Web page that the client software application is embedded within, except if the client software application was not embedded in a Web page. If the client software application is not embedded in a Web page, but an Active Stream Redirector (ASX) file (for more information, see [\[MSDN-WMMETA\]](#)) was obtained from a Web page, then this field SHOULD be set to the URL to that Web page.

If none of the above applies, this field MUST be set to "-".

The syntax of the **cs-Referer** field is defined as follows:

```
cs-Referer= "-"  
           / URI-reference ; as defined in section 4.1 of [RFC3986]
```

Examples:

```
http://www.adventure-works.com/default.htm
```

2.1.32 cs-url

This field MUST specify the URL for the streaming content originally requested by the client.

Note that the value of this field can be different from the URL actually used if the server redirected the client to a different URL, or if the client decided to use a streaming protocol that is different from the one indicated by the URL scheme of the original URL.

When the [MSB Protocol](#) is used, the "asfm" MUST be used as the URL scheme in the **cs-url** field.

The syntax of the **cs-url** field is defined as follows:

```
cs-url= URI-reference; as defined in section 4.1 of [RFC3986].
```

Example 1:

```
mms://www.adventure-works.com/some/content.asf
```

Example 2:

```
asfm://239.1.2.3:9000
```

2.1.33 cs-uri-stem

This field MUST specify the URL actually used by the client. Any query strings MUST be excluded from the URL. (This means that the value of the **cs-uri-stem** field is equal to the URL actually used by the client, truncated at the first "?" character.)

Note that the value of this field can be different from the URL originally requested by the client if the server redirected the client to a different URL, or if the client decided to use a streaming protocol that is different from the one indicated by the URL scheme of the original URL.

When the [Media Stream Broadcast \(MSB\) Protocol](#) is used (for more information, see [MS-MSB]), the "asfm" MUST be used as the URL scheme in the **cs-uri-stem** field.

The syntax of the **cs-uri-stem** field is defined as follows:

```
cs-uri-stem= URI-reference; as defined in section 4.1 of [RFC3986].
```

Example:

```
rtspt://server/test/sample.asf
```

2.1.34 cs-User-Agent

The purpose of this field is to specify information regarding the client application that is sending the logging message.

The **cs-User-Agent** field SHOULD be set to the same value that Internet Explorer specifies on the User-Agent HTTP protocol header. The field MAY be set differently as long as it adheres to the ABNF syntax as shown in the code example below.

If a logging message is forwarded by a proxy, the **cs-User-Agent** field MUST begin with the string "_via_". The original value specified by the client (which may be another proxy) on the **cs-User-Agent** field SHOULD be discarded. The proxy SHOULD include a product token on the **cs-User-Agent** field that specifies the brand and version of the proxy.

The syntax of the **cs-User-Agent** field is defined as follows:

```
cs-User-Agent= [ "_via_HTTP/1.0_" ]  
1*( product; [RFC2616] section 3.8  
| comment ); [RFC2616] section 2.2
```

Example 1: media player embedded in Internet Explorer 6 on Windows XP SP2:

```
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1;_SV1)
```

Example 2: application based on Windows Media Format 9 Series SDK:

```
Application/2.3 (WMFSDK/9.0.1234)
```

Example 3: proxy:

```
_via_HTTP/1.0_WMCaCheProxy/9.00.00.1234
```

2.1.35 cs-user-name

This field MUST be set to "-".

The syntax of the **cs-user-name** field is defined as follows:

```
cs-user-name= "-"
```

Example:

```
-
```

2.1.36 date

This field MUST specify the current date on the client when the log message is created. The time MUST be specified in UTC.

The syntax of the **date** field is defined as follows:

```
date= date-year "-" date-month "-" date-day
```

Example:

```
1997-10-09
```

2.1.37 filelength

This field MUST specify the length of the **ASF** file, in seconds. For a live broadcast stream, the value for **filelength** is undefined and MUST be set to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **filelength** field is defined as follows:

```
filelength= 1*10DIGIT
```

Example:

```
60
```

2.1.38 filesize

This field MUST specify the size of the ASF file, in bytes. For a live broadcast stream, the value for the **filesize** field is undefined and MUST be set to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **filesize** field is defined as follows:

filesize= 1*10DIGIT

Example:

86000

2.1.39 protocol

This field MUST specify the protocol used to stream content to the client.

If the [Windows Media HTTP Streaming Media Protocol](#) was used, the value of the **protocol** field MUST be "http".

If the [RTSP Windows Media Extensions](#) was used, and all ASF data packets were transmitted over TCP, the value of the **protocol** field MUST be "rtspt". If some ASF data packets were transmitted over UDP, the value of the **protocol** field MUST be "rtspu".

If the [MSB Protocol](#) was used, the value of the **protocol** field MUST be "asfm".

Note The value for **protocol** can be different from the URL moniker used in the stream request.

The syntax of the **protocol** field is defined as follows:

```
protocol= "http" / "rtspt" / "rtspu" / "asfm"
```

Example:

http

2.1.40 s-content-path

This field MUST be set to "-".

The syntax of the **s-content-path** field is defined as follows:

```
s-content-path = "-"
```

Example:

-

2.1.41 s-cpu-util

When a client creates a logging message, it MUST specify the **s-cpu-util** field as "-".

If a proxy is forwarding the logging message on behalf of a client (which may be another proxy), the proxy MUST replace the value of the **s-cpu-util** field that was specified by the client with the proxy's current CPU load, in percentage, at the time of forwarding the logging message. If the proxy uses

symmetric multi-processing, the CPU load value MUST be calculated as the average for all processors.

When a numerical value is specified, the value MUST be an integer in the range from 0 through 100.

The syntax of the **s-cpu-util** field is defined as follows:

```
s-cpu-util = "-" | 1*2DIGIT | "100"
```

Example:

```
40
```

2.1.42 s-dns

This field SHOULD specify the DNS name of the proxy if a proxy is forwarding the logging message on behalf of a client (which may be another proxy). The proxy MUST replace the value of the **s-dns** field that was specified by the client with its own DNS name or with "-" if the DNS name cannot be determined.

When a client creates a logging message, it SHOULD specify the **s-dns** field as "-" but MAY specify the DNS name of the server that the clientstreamed the content from.

The syntax of the **s-dns** field is defined as follows:

```
s-dns= "-"  
      | reg-name ; as defined in [RFC3986].
```

Example:

```
wmt.adventure-works.com
```

2.1.43 s-ip

For Legacy and Streaming Logs, this field MUST specify the IP address of the server that the client streamed the content from.

For Rendering Logs, the field MUST specify the IP address of the proxy if a proxy is forwarding the logging message on behalf of a client. The proxy MUST replace the value of the **s-ip** field that was specified by the client (which may be another proxy) with the IP address used by the proxy when forwarding the Rendering Log to the server (which may be another proxy).

When a client creates a rendering log, it SHOULD specify the **s-ip** field as "-" but can specify the IP address of the server that the clientstreamed the content from.

The syntax of the **s-ip** field is defined as follows:

```
s-ip = "-" | ip_addr
```

Example:

```
155.12.1.234
```

2.1.44 s-pkts-sent

This field MUST be set to "-".

The syntax of the **s-pkts-sent** field is defined as follows:

```
s-pkts-sent= "-"
```

Example:

```
-
```

2.1.45 s-proxied

This field MUST be set to "1" in a logging message that is being forwarded by a proxy. The client that creates the logging message MUST set the field to "0" and the proxy MUST change the value to "1" when it forwards the logging message.

The syntax of the **s-proxied** field is defined as follows:

```
s-proxied= "0" / "1"
```

Example:

```
1
```

2.1.46 s-session-id

This field MUST be set to "-".

The syntax of the **s-session-id** field is defined as follows:

```
s-session-id= "-"
```

Example:

```
-
```

2.1.47 s-totalclients

When a client creates a logging message, it MUST specify the **s-totalclients** field as "-".

If a proxy is forwarding the logging message on behalf of a client (which may be another proxy), then the proxy MUST replace the value of the **s-totalclients** field that was specified by the client with the total number of clients connected to the proxy server (for all target servers combined).

When a numerical value is specified, the value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **s-totalclients** field is defined as follows:

```
s-totalclients = "-" | 1*10DIGIT
```

Example:

```
201
```

2.1.48 sc-bytes

This field MUST be set to "-".

The syntax of the **sc-bytes** field is defined as follows:

```
sc-bytes= "-"
```

Example:

```
-
```

2.1.49 time

This field MUST specify the current time on the client when the log message is created. The time MUST be specified in UTC.

The syntax of the **time** field is defined as follows:

```
time= time-hour ":" time-min ":" time-sec
```

Example:

```
15:30:30
```

2.1.50 transport

This field MUST specify the transport protocol used to receive the ASF data packets.

The syntax of the **transport** field is defined as follows:

```
transport= "UDP" | "TCP"
```


Example:

```
UDP
```

2.1.51 videocodec

This field SHOULD specify a list of video codecs that are used to decode the video streams accessed by the client. Each codec MUST be listed only once, regardless of the number of streams decoded by that codec.

The value for **videocodec** MUST NOT exceed 256 characters in total length.

The syntax of the **videocodec** field is defined as follows:

```
codec-name= *VCHAR  
videocodec= "-" | ( codec-name *( ";" codec-name ) )
```

Example:

```
Microsoft_MPEG-4_Video_Codec_V2
```

2.1.52 x-duration

For Legacy and Rendering Log messages, this field MUST specify how much of the content has been rendered (played) to the end user, specified in seconds. Time spent buffering data MUST NOT be included in this value.

Playback at non-normal play speed does not affect the amount of content rendered, when expressed in time units. For example, if the client was rewinding the content, the **x-duration** value can be computed as the absolute value of the difference between the starting presentation time and ending presentation time.

For Streaming Log messages, the **x-duration** field MUST specify the time it took to receive the content, in seconds.

Fractional time amounts MUST be rounded to the nearest larger integer value.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **x-duration** field is defined as follows:

```
x-duration= 1*10DIGIT
```

Example:

```
31
```

2.2 Logging Message: W3C Syntax

A World Wide Web Consortium (W3C) format logging message consists of the values of various fields, each value separated from the next by a single space character. Logging messages that adhere to this syntax are said to use the W3C format because the syntax is conformant with the syntax for logging entries in the Extended Log File Format (for more information, see [\[W3C-EXLOG\]](#)), which is defined by W3C.

Section [2.2.1](#) specifies the W3C format syntax used in most logging messages. Section [2.2.2](#) specifies the W3C format syntax used in certain Rendering log messages, and section [2.2.3](#) specifies the W3C format syntax used in Connect-time log messages.

The sections mentioned above define the ordering of the fields in the W3C format syntax but not how the values of the fields are assigned. The rules governing the values of the individual fields depend on the logging message in which the W3C format syntax is used. For example, the [s-ip](#) field is used as defined in section [2.1.43](#) for some logging messages, while other logging messages provide an alternate definition of the **s-ip** field.

All W3C format syntax MUST use the UTF-8 character set as specified in [\[RFC3629\]](#). In any fields that specify a URL, such as [cs-url](#), the URL MUST be encoded using percent-encoding, as specified in [\[RFC3986\]](#) section 2.1.

A single dash character (which is represented by U+002D and by "-" in ABNF syntax) MUST be used to indicate that the value is empty — that is, it is either not available or not applicable.

All spaces embedded within a field value MUST be replaced by an underscore character (which is represented by U+005F and by "_" in ABNF syntax). For example, "MPEG Layer-3" would be transformed into "MPEG_Layer-3" in a W3C-format logging message.

Note Transformations defined in this section are not necessarily reversible. Methods for parsing, analyzing, or extracting information from logging messages are implementation-specific and are outside the scope of this specification.

2.2.1 Basic Logging Syntax

Most logging messages contain logging information in W3C format, adhering to the syntax specified below. The logging information consists of either 44 or 47 fields.

```
log_data44 = c-ip SP date SP time SP c-dns SP cs-uri-stem SP c-starttime SP
           x-duration SP c-rate SP c-status SP c-playerid SP
           c-playerversion SP c-playerlanguage SP cs-User-Agent SP
           cs-Referer SP c-hostexe SP c-hostexever SP c-os SP c-osversion SP
           c-cpu SP filelength SP filesize SP avgbandwidth SP protocol SP
           transport SP audiocodec SP videocodec SP c-channelURL SP sc-bytes SP
           c-bytes SP s-pkts-sent SP c-pkts-received SP c-pkts-lost-client SP
           c-pkts-lost-net SP c-pkts-lost-cont-net SP c-resendreqs SP
           c-pkts-recovered-ECC SP c-pkts-recovered-resent SP c-buffercount SP
           c-totalbuffertime SP c-quality SP s-ip SP s-dns SP
           s-totalclients SP s-cpu-util
           [ SP cs-url SP cs-media-name SP cs-media-role ]
```

2.2.2 Extended Logging Syntax

Certain types of "rendering" log messages (section [2.7](#)) contain logging information in the W3C format defined below. This logging information consists of 52 fields:

```

log_data52 = c-ip SP date SP time SP c-dns SP cs-uri-stem SP c-starttime SP
x-duration SP c-rate SP c-status SP c-playerid SP
c-playerversion SP c-playerlanguage SP cs-User-Agent SP
cs-Referer SP c-hostexe SP c-hostexever SP c-os SP c-osversion SP
c-cpu SP filelength SP filesize SP avgbandwidth SP protocol SP
transport SP audiocodec SP videocodec SP c-channelURL SP sc-bytes SP
c-bytes SP s-pkts-sent SP c-pkts-received SP c-pkts-lost-client SP
c-pkts-lost-net SP c-pkts-lost-cont-net SP c-resendreqs SP
c-pkts-recovered-ECC SP c-pkts-recovered-resent SP c-buffercount SP
c-totalbuffertime SP c-quality SP s-ip SP s-dns SP
s-totalclients SP s-cpu-util SP cs-user-name SP s-session-id SP
s-content-path SP cs-url SP cs-media-name SP c-max-bandwidth SP
cs-media-role SP s-proxied

```

2.2.3 Connect-Time Logging Syntax

Connect-time log messages (section [2.8](#)) contain logging information in the W3C format defined below. This logging information consists of eight fields.

```

log_data8 = c-dns SP c-ip SP c-os SP c-osversion SPdate SP time SP
c-cpu SP transport

```

2.3 Logging Messages Sent to Web Servers

Most of the logging messages defined in this specification can be sent to a HTTP Web server. The URL for the HTTP Web server for which logging messages are submitted can be specified in an Active Stream Redirector (ASX) file (for more information, see [\[MSDN-WMMETA\]](#)). Some of the compatible streaming protocols (listed in section [1.3](#)) can also specify the HTTP Web server URL through mechanisms that are specific to the streaming protocol. The syntax for the logging URL is defined as follows:

```

log-URL = Request-URI

```

The resource that is identified by log-URL MUST be capable of accepting and responding to the HTTP GET and POST request methods described in this section; however, the methods for doing so are implementation-specific.

Prior to sending a logging message to a Web server, a client SHOULD send an HTTP GET request to the specified Web server URL to validate the URL. The logging validation request MUST adhere to the following ABNF syntax:

```

web-server-validate = "GET" SP log-URL SP HTTP-Version CRLF
*( VCHAR /CLRF )

```

The web server's response MUST adhere to the following ABNF syntax:

```

web-server-validate-response = HTTP-Version "200 OK" CRLF

```

```

*( VCHAR / CRLF ) "<body><h1>"
( "NetShow ISAPI Log Dll" /
( "WMS ISAPI Log Dll/"
1*4DIGIT "." 1*4DIGIT "." 1*4DIGIT "." 1*4DIGIT ) )
*( VCHAR / CRLF ) "</h1>" *( VCHAR / CRLF ) </body>" *( VCHAR / CRLF )

```

The client SHOULD send the logging message to the Web server if the server's response adheres to the syntax for web-server-validate-response, above. If the client sent a request to validate the URL, and the server's response does not adhere to the syntax for web-server-validate-response, then this might mean that the URL is invalid. In this case, the client SHOULD NOT send the logging message.

When sending the logging message, the client MUST include the logging message in the body of a HTTP POST request.

All logging message requests that are sent to a Web server MUST adhere to the following ABNF syntax:

```

web-server-request = "POST" SP log-URL SP HTTP-Version CRLF
                    *( VCHAR / CRLF )
                    web-server-log

```

The logging message sent in the web-server-request message body MUST adhere to the following ABNF syntax:

```

web-server-log = "MX_STATS_LogLine:" SP TAB
                log_data44; defined in section 2.2.1

```

All HTTP GET and POST requests sent by the client or Web server must be syntactically correct as per [RFC1945](#) or [RFC2616](#). Any header or content element not explicitly represented in one of the preceding ABNF syntax examples MUST be ignored by the recipient.

For an example of logging URL validation and the subsequent transmission of a logging message to a Web server, see section [3.6](#).

2.4 Logging Message: XML Schema

Logging messages can be represented in XML. This section defines the schema used by all logging messages for which an XML representation has been defined with the exception of the [Connect-Time Log](#). The XML scheme for the Connect-Time Log is defined in section [2.8](#).

The XML-format log embeds W3C-format logging information inside the "Summary" XML tag. Individual logging fields are also represented using their own XML tags.

If the entity that generates the XML-format logging message (that is, the client) has access to a Content Description, then each name/value pair in the Content Description SHOULD be encoded as shown by the "contentdescription" syntax element in the ABNF syntax as shown in the code example below.

The Content Description is a data structure that is provided by Windows Media Services. If no Content Description is available to the client, then the "contentdescription" syntax element MUST NOT be included in the XML-format logging message.

If the entity that generates the XML-format logging message (that is, the client) submits additional or custom logging information, then it SHOULD be encoded as shown by the "client-logging-data" syntax element in the ABNF syntax below. For an example illustrating submission of custom logging information, see section [3.2](#).

If no additional logging information is available, the "client-logging-data" syntax element MUST NOT be included in the XML-format logging message.

The XML-format logging syntax is defined using ABNF as shown in the code example below. Although not explicitly shown by the syntax, linear white space, including CR LF sequences, is allowed on each side of XML tags.

```
xml-tag = 1*ALPHA
cd-name = xml-tag
cd-value = xml-tag
cd-name-value-pair = "<" cd-name ">"
cd-value
    "</" cd-name ">"

contentdescription = "<ContentDescription>"
    *cd-name-value-pair
    "</ContentDescription>"

client-logging-data = "<" xml-tag ">"
    *cdl-name-value-pair
    "</" xml-tag ">"

xml-log = "<XML>"
    "<Summary>" summary-log "</Summary>"
    "<c-ip>" "0.0.0.0" "</c-ip>"
    "<date>" date "</date>"
    "<time>" time "</time>"
    "<c-dns>" c-dns "</c-dns>"
    "<cs-uri-stem>" cs-uri-stem "</cs-uri-stem>"
    "<c-starttime>" c-starttime "</c-starttime>"
    "<x-duration>" x-duration "</x-duration>"
    "<c-rate>" c-rate "</c-rate>"
    "<c-status>" c-status "</c-status>"
    "<c-playerid>" c-playerid "<c-playerid>"
    "<c-playerversion>" c-playerversion "</c-playerversion>"
    "<c-playerlanguage>" c-playerlanguage "</c-playerlanguage>"
    "<cs-User-Agent>" cs-User-Agent "</cs-User-Agent>"
    "<cs-Referer>" cs-Referer "<cs-Referer>"
    "<c-hostexe>" c-hostexe "</c-hostexe>"
    "<c-hostexeever>" c-hostexeever "</c-hostexeever>"
    "<c-os>" c-os "</c-os>"
    "<c-osversion>" c-osversion "</c-osversion>"
    "<c-cpu>" c-cpu "</c-cpu>"
    "<filelength>" filelength "</filelength>"
    "<filesize>" filesize "</filesize>"
    "<avgbandwidth>" avgbandwidth "</avgbandwidth>"
    "<protocol>" protocol "</protocol>"
    "<transport>" transport "</transport>"
    "<audiocodec>" audiocodec "</audiocodec>"
    "<videocodec>" videocodec "</videocodec>"
    "<c-channelURL>" c-channelURL "</c-channelURL>"
    "<sc-bytes>" sc-bytes "</sc-bytes>"
    "<c-bytes>" c-bytes "</c-bytes>"
```

```

"<s-pkts-sent>" s-pkts-sent "</s-pkts-sent>"
"<c-pkts-received>" c-pkts-received "</c-pkts-received>"
"<c-pkts-lost-client>" c-pkts-lost-client "</c-pkts-lost-client>"
"<c-pkts-lost-net>" c-pkts-lost-net "</c-pkts-lost-net>"
"<c-pkts-lost-cont-net>" c-pkts-lost-cont-net "</c-pkts-lost-cont-net>"
"<c-resendreqs>" c-resendreqs "</c-resendreqs>"
"<c-pkts-recovered-ECC>" c-pkts-recovered-ECC "</c-pkts-recovered-ECC>"
"<c-pkts-recovered-resent>" c-pkts-recovered-resent "</c-pkts-recovered-resent>"
"<c-buffercount>" c-buffercount "</c-buffercount>"
"<c-totalbuffertime>" c-totalbuffertime "</c-totalbuffertime>"
"<c-quality>" c-quality "</c-quality>"
"<s-ip>" "-" "</s-ip>"
"<s-dns>" "-" "</s-dns>"
"<s-totalclients>" "-" "</s-totalclients>"
"<s-cpu-util>" "-" "</s-cpu-util>"
"<cs-url>" cs-url "</cs-url>"
[ contentdescription ]
*client-logging-data
"</XML>"

```

The syntax only defines the ordering of the fields and the XML tag assigned to each field; it does not define how the values of the fields are assigned. The rules governing the values of the individual fields depend on the logging message in which the XML-format syntax is used.

The XML-format logging syntax MUST use the UTF-8 character set, as specified in [\[RFC3629\]](#). In any fields that specify a URL, such as cs-url, the URL MUST be encoded using percent-encoding, as specified in [\[RFC3986\]](#) section 2.1.

A single dash character (which is represented by U+002D and by "-" in ABNF syntax) MUST be used to indicate that the value is empty — that is, it is either not available or not applicable.

All spaces embedded within a field value MUST be replaced by an underscore character (which is represented by U+005F and by "_" in ABNF syntax). For example, "MPEG Layer-3" would be transformed into "MPEG_Layer-3" in a W3C-format logging message.

2.5 Legacy Log

The Legacy Log is also called a combination log because it contains both rendering and streaming information. The Legacy Log can be either in W3C format or XML format. A Legacy Log can be sent either to Windows Media Services or to a Web server.

2.5.1 Common Definitions

The following ABNF syntax rules applies to all variants of the legacy log: [<4>](#)

```

s-cpu-util      = "-"
c-ip            = "0.0.0.0"
s-dns          = "-"

```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **audiocodec**
- **avgbandwidth**
- **c-buffercount**

- **c-channelURL**
- **c-cpu**
- **c-dns**
- **c-hostexe**
- **c-hostexever**
- **c-os**
- **c-osversion**
- **c-pkts-lost-client**
- **c-pkts-lost-cont-net**
- **c-pkts-lost-net**
- **c-pkts-recovered-ECC**
- **c-pkts-recovered-resent**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-quality**
- **c-rate**
- **c-resendreqs**
- **c-starttime**
- **c-status**
- **c-totalbuffertime**
- **cs-Referer**
- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **protocol**

- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **transport**
- **videocodec**
- **x-duration**

The [Legacy Log](#) SHOULD include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.[<5>](#)

2.5.2 Legacy Log in W3C Format

The ABNF syntax for a [Legacy Log](#) in W3C format that is sent to Windows Media Services is defined as follows:

```
legacy-log-W3C      = log_data44          ; defined in section 2.2.1
s-ip                = "-"
```

2.5.3 Legacy Log in XML Format

The ABNF syntax for a [Legacy Log](#) in XML format that is sent to Windows Media Services is defined as follows:[<6>](#)

```
legacy-log-XML     = xml-log            ; defined in section 2.4
summary-log        = log_data44        ; defined in section 2.2.1
s-ip               = "-"
```

2.5.4 Legacy Log Sent to a Web Server

The ABNF syntax for a [Legacy Log](#) that is submitted to a Web server is defined as follows:

```
legacy-web-server-log = web-server-log    ; defined in section 2.3
```

The value of the **s-ip** field MUST be assigned as defined in section [2.1.43](#).

2.6 Streaming Log

The Streaming Log specifies how the client received streaming data but not how the client rendered the data. A Streaming Log can be sent either to Windows Media Services or to a Web server.

2.6.1 Common Definitions

The following ABNF syntax rules applies to all variants of the [Streaming Log](#):

```
audiocodec      = "-"
c-ip            = "0.0.0.0"
s-cpu-util     = "-"
s-dns          = "-"
videocodec     = "-"
```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **avgbandwidth**
- **c-buffercount**
- **c-channelURL**
- **c-cpu**
- **c-dns**
- **c-hostexex**
- **c-hostexever**
- **c-os**
- **c-osversion**
- **c-pkts-lost-client**
- **c-pkts-lost-cont-net**
- **c-pkts-lost-net**
- **c-pkts-recovered-ECC**
- **c-pkts-recovered-resent**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-quality**
- **c-rate**
- **c-resendreqs**
- **c-starttime**
- **c-status**
- **c-totalbuffertime**

- **cs-Referer**
- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **protocol**
- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **transport**
- **x-duration**

The Streaming Log MUST include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.

2.6.2 Streaming Log Sent to Windows Media Services

The [Streaming Log](#) sent to Windows Media Services is in XML format and MUST adhere to the following ABNF syntax: [<7>](#)

```
streaming-log      = xml-log           ; defined in section 2.4
summary-log       = log_data44        ; defined in section 2.2.1
s-ip              = "-"
```

2.6.3 Streaming Log Sent to a Web Server

The ABNF syntax for a [Streaming Log](#) that is submitted to a Web server is defined as follows:

```
streaming-web-server-log = web-server-log; defined in section 2.3
```

The value of the **s-ip** field MUST be assigned as specified in section [2.1.43](#).

2.7 Rendering Log

The Rendering Log describes playback of content by a client and is submitted to the upstream origin server (or a configured proxy) when the client ends playback. A Rendering Log can be sent either to Windows Media Services or to a Web server.

2.7.1 Common Definitions

The following ABNF syntax rules apply to all variants of the [Rendering Log](#):

```
avgbandwidth           = "-"
c-buffercount          = "-"
c-pkts-lost-client     = "-"
c-pkts-lost-cont-net  = "-"
c-pkts-lost-net       = "-"
c-pkts-received       = "-"
c-pkts-recovered-ECC  = "-"
c-pkts-recovered-resent = "-"
c-quality              = "100"
c-resendreqs          = "-"
c-totalbuffertime     = "-"
protocol               = "Cache"
transport              = "-"
```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **audiocodec**
- **c-channelURL**
- **c-cpu**
- **c-hostexe**
- **c-hostexevers**
- **c-ip**
- **c-os**
- **c-osversion**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-rate**
- **c-starttime**
- **c-status**
- **cs-Referer**

- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **s-cpu-util**
- **s-dns**
- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **videocodec**
- **x-duration**

The Rendering Log MUST include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.

2.7.2 Rendering Log Sent to Windows Media Services

The [Rendering Log](#) sent to Windows Media Services is in XML format and MUST adhere to the following ABNF syntax:

```
rendering-log      = xml-log           ; defined in section 2.4
summary-log       = log_data52       ; defined in section 2.2.2
```

The values of the following fields MUST be assigned as defined in section [2.1](#): **c-max-bandwidth**, **cs-user-name**, **s-content-path**, **s-ip**, **s-proxied**, and **s-session-id**.

2.7.3 Rendering Log Sent to a Web Server

The ABNF syntax for a [Rendering Log](#) that is submitted to a Web server is defined as follows:

```
rendering-web-server-log = web-server-log; defined in section 2.3
```

The value of the **c-ip** field MUST be assigned as defined in section [2.1.8](#). The value of the **s-ip** field MUST be assigned as defined in section [2.1.43](#).

2.8 Connect-Time Log

The purpose of the Connect-Time Log is to specify some minimal amount of logging information about the client. It can be useful in cases where a client starts to stream some content but is disconnected from the network before it has the opportunity to create a [Streaming Log](#).

If a client sends a Connect-Time Log to the server at the start of the streaming **session**, the Connect-Time Log ensures that the server has received at least this minimal logging information in the case where the client subsequently is disconnected from the network.

Connect-Time Logs are not defined for Web servers. Connect-Time Logs are only defined in XML format, and the ABNF syntax is as follows:

```
connect-time-log = "<XML>"
  "<Summary>"
    log_data8 ; defined in section 2.2.3
  "</Summary>"
  "<c-dns>" c-dns "</c-dns>"
  "<c-ip>" c-ip "</c-ip>"
  "<c-os>" c-os "</c-os>"
  "<c-osversion>" c-osversion "</c-osversion>"
  "<date>" date "</date>"
  "<time>" time "</time>"
  "<c-cpu>" c-cpu "</c-cpu>"
  "<transport>" transport "</transport>"
"</XML>"

c-ip = "0.0.0.0"
```

The values of the following fields **MUST** be assigned as defined in section [2.1](#) :

- **c-dns**
- **c-os**
- **c-osversion**
- **date**
- **time**
- **c-cpu**
- **transport**

3 Structure Examples

3.1 Legacy Logging Message

The following is an example of a legacy logging message in W3C format:

```
0.0.0.0 2003-09-27 00:27:24 - http://10.194.20.175/mcast1200K 0 42 1
200 {3300AD50-2C39-46c0-AE0A-B4C904C7848E} 9.0.0.2980 en-US
WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008 - wmplayer.exe 9.0.0.2980
Windows_XP 5.1.0.2600 Pentium 1801 268885194 1255347 http TCP
Windows_Media_Audio_9 Windows_Media_Video_9 - - 6321233 - 4496 0 0 0 0
0 0 1 0 100 - - - -
```

The following is an example of a legacy logging message in XML format:

```
<XML>
<Summary>0.0.0.0 2003-09-27 00:27:24 - http://10.194.20.175/mcast1200K 0 42 1 200
{3300AD50-2C39-46c0-AE0A-B4C904C7848E} 9.0.0.2980
en-US WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008 - wmplayer.exe 9.0.0.2980
Windows_XP 5.1.0.2600 Pentium 1801 268885194 1255347
http TCP Windows_Media_Audio_9 Windows_Media_Video_9
- - 6321233 - 4496 0 0 0 0 0 0 1 0 100 - - - -
http://10.194.20.175/mcast1200K?WMBitrate=6000000 30MinTV_1200k_1s_1s_0Q.wmv -
</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2003-09-27</date>
<time>00:27:24</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://10.194.20.175/mcast1200K</cs-uri-stem>
<c-starttime>0</c-starttime>
<x-duration>42</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-B4C904C7848E}</c-playerid>
<c-playerversion>9.0.0.2980</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-Agent>WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008</cs-User-Agent>
<cs-Referer>-</cs-Referer>
<c-hostexe>wmplayer.exe</c-hostexe>
<c-hostexeever>9.0.0.2980</c-hostexeever>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>1801</filelength>
<filesize>268885194</filesize>
<avgbandwidth>1255347</avgbandwidth>
<protocol>http</protocol>
<transport>TCP</transport>
<audiocodec>Windows_Media_Audio_9</audiocodec>
<videocodec>Windows_Media_Video_9</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>6321233</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>4496</c-pkts-received>
<c-pkts-lost-client>0</c-pkts-lost-client>
<c-pkts-lost-net>0</c-pkts-lost-net>
<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>
<c-resendreqs>0</c-resendreqs>
<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>
```

```

<c-pkts-recovered-resent>0</c-pkts-recovered-resent>
<c-buffercount>1</c-buffercount>
<c-totalbuffertime>0</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>http://10.194.20.175/mcast1200K?WMBitrate=6000000</cs-url>
<ContentDescription>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL>30MinTV_1200k_1s_1s_0Q.wmv</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL>
<WMS_CONTENT_DESCRIPTION_COPIED_METADATA_FROM_PLAYLIST_FILE>1</WMS_CONTENT_DESCRIPTION_COPIED_METADATA_FROM_PLAYLIST_FILE>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_DURATION>1800501</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_DURATION>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_START_OFFSET>1450</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_START_OFFSET>
<WMS_CONTENT_DESCRIPTION_SERVER_BRANDING_INFO>WMServer/9.0</WMS_CONTENT_DESCRIPTION_SERVER_BRANDING_INFO>
</ContentDescription>
</XML>

```

The following is an example of how a legacy log may appear as sent to a Web server:

```

MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mmsu://foo.microsoft.com/testfile.wma 30 1 1 200 {35301A88-93D3-4F3A-
A284-30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938
Windows_2000 5.0.0.2195 Pentium 225 4551684 1528 mms UDP - - - - 29868
- 4 0 0 0 0 0 0 0 100 172.29.237.102 - - -

```

3.2 Defining Custom Namespaces in an XML Log

An Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)) can be used to append log data to the XML log structure. Vendors may define any number of custom namespaces and name-value pairs in the "client-logging-data" structure, as specified in section [2.4](#), following the Content Description structure.

The following example illustrates how to add the **cs-media-role** field (section [2.1.30](#)) by using an .asx file:

```

<ASX version="3.0">
  <ENTRY>
    <TITLE> My Title </TITLE>
    <Author> My Author </Author>
    <PARAM name="log:cs-media-role" value="Advertisement" />
    <REF href="http://www.foo.MyDomain.com/live" />
  </ENTRY>
</ASX>

```

The additional and/or custom logging information is specified through the use of the PARAM element. To use the PARAM element in this way, the NAME attribute is set to "log:" followed by a log field name and a corresponding VALUE attribute. The log field specified in the NAME attribute is set

to the value of the VALUE attribute. If the log does not already contain a field with the specified name, it will be added.

An XML namespace has to be defined for each custom log field specified in an .asx file. This namespace is appended to the NAME attribute and is separated from the field name by a second colon (":"). Because everything after the second colon is treated as a namespace, the field name should not contain a colon.

The following example illustrates the specification of custom log fields using an .asx file:

```
<ASX version="3.0">
  <ENTRY>
    <TITLE> My Title </TITLE>
    <Author> My Author </Author>
    <PARAM name="log:vendor-field1:VendorNameSpace" value="Value1" />
    <PARAM name="log:vendor-field2:VendorNameSpace" value="Value2" />
    <REF href="http://www.foo.MyDomain.com/live" />
  </ENTRY>
</ASX>
```

When an XML log is sent to a server for this .asx file, the new namespace is inserted after the Content Description section, as shown in the following example (many log fields extraneous to this example have been omitted for brevity and clarity):

```
<XML>
  <Summary>0.0.0.0 2003-09-27 00:27:24 ... </Summary>
  <c-ip>0.0.0.0</c-ip>
  <date>2003-09-27</date>
  <time>00:27:24</time>
  ...
  <ContentDescription>
  ...
</ContentDescription>
  <VendorNameSpace>
    <vendor-field1>Value1</vendor-field1>
    <vendor-field2>Value2</vendor-field2>
  </VendorNameSpace>
</XML>
```

3.3 Example Streaming Log Messages

The following is an example of a [Streaming Log](#) in XML format:

```
<XML>
<Summary>0.0.0.0 2006-05-01 21:34:01 -
http://foo.microsoft.com/content.wmv 4 0 1 200 {3300AD50-2C39-46c0-
AE0A-3E0B6EFB86DC} 10.0.0.3802 en-US
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_(WMFSDK/10.0.0.3802)
_WMPPlayer/10.0.0.4019 http://bar.microsoft.com iexplore.exe
6.0.2900.2180 Windows_XP 5.1.0.2600 Pentium 130 638066 - http TCP - - -
-0 - 0 0 0 0 0 0 0 0 0 0 100 - - - -
```



```

http://foo.microsoft.com/content.wmv - -</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2006-05-01</date>
<time>21:34:01</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://foo.microsoft.com/content.wmv</cs-uri-stem>
<c-starttime>4</c-starttime>
<x-duration>0</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-3E0B6EFB86DC}</c-playerid>
<c-playerversion>10.0.0.3802</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-
Agent>Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)__(WMFSDK/10.0.0.3802)_WMPlayer/10
.0.0.4019</cs-User-Agent>
<cs-Referer>http://bar.microsoft.com</cs-Referer>
<c-hostexe>iexplore.exe</c-hostexe>
<c-hostexever>6.0.2900.2180</c-hostexever>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>130</filelength>
<filesize>638066</filesize>
<avgbandwidth>-</avgbandwidth>
<protocol>http</protocol>
<transport>TCP</transport>
<audiocodec>-</audiocodec>
<videocodec>-</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>0</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>0</c-pkts-received>
<c-pkts-lost-client>0</c-pkts-lost-client>
<c-pkts-lost-net>0</c-pkts-lost-net>
<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>
<c-resendreqs>0</c-resendreqs>
<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>0</c-pkts-recovered-resent>
<c-buffercount>0</c-buffercount>
<c-totalbuffertime>0</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>http://foo.microsoft.com/content.wmv</cs-url>
<cs-media-name>-</cs-media-name>
<cs-media-role>-</cs-media-role>
</XML>

```

The following is an example of how a Streaming Log may appear as sent to a Web server:

```

MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mmsu://foo.microsoft.com/testfile.wma 30 1 1 200 {35301A88-93D3-4F3A-
A284-30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938
Windows_2000 5.0.0.2195 Pentium 225 4551684 1528 mms UDP - - - - 29868
- 4 0 0 0 0 0 0 0 100 172.29.237.102 - - -
mmsu://foo.microsoft.com/testfile.wma - -

```

3.4 Example Rendering Log Messages

The following is an example of a [Rendering Log](#) in XML format:

```
<XML>
<Summary>0.0.0.0 2006-05-01 21:34:01 -
http://foo.microsoft.com/content.wmv 4 0 1 200 {3300AD50-2C39-46c0-
AE0A-3E0B6EFB86DC} 10.0.0.3802 en-US
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_
(WMFS DK/10.0.0.3802)_WMPlayer/10.0.0.4019 http://bar.microsoft.com iexplore.exe
6.0.2900.2180 Windows_XP 5.1.0.2600 Pentium 130 638066 - Cache -
Windows_Media_Audio_9 Windows_Media_Video_9 - - 0 - - - - - - - - - -
100 - - - - - http://foo.microsoft.com/content.wmv - - - 0
</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2006-05-01</date>
<time>21:34:01</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://foo.microsoft.com/content.wmv</cs-uri-stem>
<c-starttime>4</c-starttime>
<x-duration>0</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-3E0B6EFB86DC}</c-playerid>
<c-playerversion>10.0.0.3802</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-Agent>Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_
_(WMFS DK/10.0.0.3802)_WMPlayer/10.0.0.4019</cs-User-Agent>
<cs-Referer>http://bar.microsoft.com</cs-Referer>
<c-hostexe>iexplore.exe</c-hostexe>
<c-hostexever>6.0.2900.2180</c-hostexever>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>130</filelength>
<filesize>638066</filesize>
<avgbandwidth>-</avgbandwidth>
<protocol>Cache</protocol>
<transport>-</transport>
<audiocodec>Windows_Media_Audio_9</audiocodec>
<videocodec>Windows_Media_Video_9</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>0</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>-</c-pkts-received>
<c-pkts-lost-client>-</c-pkts-lost-client>
<c-pkts-lost-net>-</c-pkts-lost-net>
<c-pkts-lost-cont-net>-</c-pkts-lost-cont-net>
<c-resendreqs>-</c-resendreqs>
<c-pkts-recovered-ECC>-</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>-</c-pkts-recovered-resent>
<c-buffercount>-</c-buffercount>
<c-totalbuffertime>-</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
```

```
<s-cpu-util>-</s-cpu-util>
<cs-url>http://foo.microsoft.com/content.wmv</cs-url>
<cs-media-name>-</cs-media-name>
<cs-media-role>-</cs-media-role>
</XML>
```

The following is an example of how a Rendering Log may appear as sent to a Web server:

```
MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mms://foo.microsoft.com/test.wma 30 1 1 200 {35301A88-93D3-4F3A-A284-
30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938 Windows_2000
5.0.0.2195 Pentium 225 4551684 1528 Cache - - - - 29868 - - - - -
- - - 100 - - - - mms://foo.microsoft.com/test.wma - -
```

3.5 Example Connect-Time Log Message

The following is an example of a [Connect-Time Log](#) message in XML format:

```
<XML>
<Summary>- 0.0.0.0 Windows 6.0.0.6000 2006-08-30 13:18:44 Pentium
TCP</Summary>
<c-dns>-</c-dns>
<c-ip>0.0.0.0</c-ip>
<c-os>Windows</c-os>
<c-osversion>6.0.0.6000</c-osversion>
<date>2006-08-30</date>
<time>13:18:44</time>
<c-cpu>Pentium</c-cpu>
<transport>TCP</transport>
</XML>
```

3.6 Example Log Sent to a Web Server

The following is an example of a client validating a logging URL and subsequently transmitting a logging message to the Web server:

```
GET /scripts/wmsiislog.dll HTTP/1.1
User-Agent: NSPlayer
Host: WebServer:8080
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Connection: close
Date: Wed, 27 Jun 2007 02:54:23 GMT
Server: Microsoft-IIS/6.0
Content-Type: text/html

<head><title>WMS ISAPI Log Dll/9.00.00.3372</title></head>
<body><h1>WMS ISAPI Log Dll/9.00.00.3372</h1></body>

POST /scripts/wmsiislog.dll HTTP/1.1
```

Content-Type: text/plain;charset=UTF-8
User-Agent: NSPlayer
Host: WebServer:8080
Content-Length: 424
Connection: Keep-Alive
Cache-Control: no-cache

```
MX_STATS_LogLine: .0.0.0.0 2007-06-27 02:52:39 - asfm://239.192.50.29:30864 0 39 1 200  
{3300AD50-2C39-46c0-AE0A-0572F2EA5330} 10.0.0.4054 en-US  
WMFSDK/10.0.0.4054 WMPlayer/10.0.0.4036 - wmplayer.exe 10.0.0.3802 Windows_XP 5.1.0.2600  
Pentium 229 10413011 411536 asfm UDP Windows_Media_Audio_9.2 -  
http://WebServer:8080/multicast.nsc - 2170350 - 182 0 0 0 0 0 0 1 3 100 239.192.50.29 - -  
- http://WebServer:8080/multicast.nsc - -
```

HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
Date: Wed, 27 Jun 2007 02:54:23 GMT
Connection: close

3.7 Parsing Windows Media Log Files

Microsoft Log Parser 2.2 is a tool that queries text-based data and other system data sources, including Windows Media log files. For more information, see [\[MSFT-LOGPARSER\]](#).

4 Security Considerations

A server that receives a logging message SHOULD validate the syntax of the fields. For example, the server should check that logging fields that are supposed to contain numerical data really do so, and that no invalid characters, such as control characters, are present. Invalid fields or characters could cause any tools that process the logging information to malfunction.

5 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

[<1> Section 2.1.5:](#) Windows Media Player 6.4 specifies the DNS name in the **c-dns** field.

[<2> Section 2.1.10:](#) On Windows Vista, **c-os** is set to "Windows".

[<3> Section 2.1.25.2:](#) Windows Media Player 6.4, Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP never specify status code 210.

[<4> Section 2.5.1:](#) Windows Media Player 6.4 specifies its own IP address in the **c-ip** field. Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP specify their own IP address in the **c-ip** field depending on the current setting of a configuration value in the user interface.

[<5> Section 2.5.1:](#) Windows Media Player 6.4, Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP never include the three optional fields.

[<6> Section 2.5.3:](#) Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista do not include the "contentdescription" and "client-logging-data" syntax elements in the XML-format logging message when using RTSP [\[MS-RTSP\]](#).

[<7> Section 2.6.2:](#) Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista do not include the "contentdescription" and "client-logging-data" syntax elements in the XML-format logging message when using RTSP [\[MS-RTSP\]](#).

6 Index

A

[Applicability](#)
[audiocodec](#)
[avgbandwidth](#)

B

[Basic logging syntax](#)

C

[c-buffercount](#)
[c-bytes](#)
[c-channelURL](#)
[c-cpu](#)
[c-dns](#)
[c-hostexe](#)
[c-hostexever](#)
[c-ip \(section 2.1.8, section 2.1.9\)](#)
[Connect-time log](#)
[Connect-time logging syntax](#)
[c-os](#)
[c-osversion](#)
[c-pkts-lost-client](#)
[c-pkts-lost-cont-net](#)
[c-pkts-lost-net](#)
[c-pkts-received](#)
[c-pkts-recovered-ECC](#)
[c-pkts-recovered-resent](#)
[c-playerid](#)
[c-playerlanguage](#)
[c-playerversion](#)
[c-quality](#)
[c-rate](#)
[c-resendreqs](#)
[cs-media-name](#)
[cs-media-role](#)
[cs-Referer](#)
[c-starttime](#)
[c-status](#)
[cs-uri-stem](#)
[cs-uri](#)
[cs-User-Agent](#)
[cs-user-name](#)
[c-totalbuffertime](#)

D

[date](#)

E

Examples
[legacy logging message example](#)
[overview](#)
[parsing Windows Media log files example](#)
[Extended logging syntax](#)

F

[Fields - vendor-extensible](#)
[filelength](#)
[filesize](#)

G

[Glossary](#)

I

[Informative references](#)
[Introduction](#)

L

Legacy log
[common definitions](#)
[overview](#)
[sent to Web server](#)
[W3C format](#)
[XML format](#)
[Legacy logging message example](#)
[Localization](#)
[Log data fields](#)
[Logging message - W3C syntax](#)
[Logging message - WXML schema](#)
[Logging message sent to Web servers](#)

N

[Normative references](#)

P

[Parsing Windows Media log files example](#)
[protocol](#)

R

References
[informative](#)
[normative](#)
[overview](#)
[Relationship to other protocols](#)
Rendering log
[common definitions](#)
[overview](#)
[sent to Web server](#)
[sent to Windows Media Services](#)

S

[sc-bytes](#)
[s-content-path](#)
[s-cpu-util](#)

[s-dns](#)
[Security](#)
[s-ip](#)
[s-pkts-sent](#)
[s-proxied](#)
[s-session-id](#)
[Status Code 200 \(No Error\)](#)
[Status Code 210 \(Client Successfully Reconnected\)](#)
[s-totalclients](#)
Streaming log
 [common definitions](#)
 [overview](#)
 [sent to Web server](#)
 [sent to Windows Media Services](#)
Structures
 [connect-time log](#)
 [legacy log](#)
 [log data fields](#)
 [logging message - W3C syntax](#)
 [logging message - XML schema](#)
 [logging message sent to Web servers](#)
 [overview](#)
 [rendering log](#)
 [streaming log](#)

T

[time](#)
[transport](#)

V

[Vendor-extensible fields](#)
[Versioning](#)
[videocodec](#)

W

[Windows behavior](#)

X

[x-duration](#)

APPENDIX B-6

[MS-WMLOG]: Windows Media Log Data Structure

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

| Date | Revision History | Revision Class | Comments |
|------------|------------------|----------------|--|
| 04/03/2007 | 0.01 | | MCPP Milestone Longhorn Initial Availability |
| 07/03/2007 | 1.0 | Major | MLonghorn+90 |
| 07/20/2007 | 2.0 | Major | Revised technical content; added example topics. |
| 08/10/2007 | 2.0.1 | Editorial | Revised and edited the technical content. |
| 09/28/2007 | 2.0.2 | Editorial | Revised and edited the technical content. |

| Date | Revision History | Revision Class | Comments |
|-------------|-------------------------|-----------------------|---|
| 10/23/2007 | 2.0.3 | Editorial | Revised and edited the technical content. |
| 11/30/2007 | 2.0.4 | Editorial | Revised and edited the technical content. |
| 01/25/2008 | 2.0.5 | Editorial | Revised and edited the technical content. |
| 03/14/2008 | 2.1 | Minor | Updated the technical content. |
| 05/16/2008 | 2.1.1 | Editorial | Revised and edited the technical content. |
| 06/20/2008 | 2.2 | Minor | Updated the technical content. |
| 07/25/2008 | 2.3 | Minor | Updated the technical content. |

Table of Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 5 |
| 1.1 | Glossary | 5 |
| 1.2 | References | 5 |
| 1.2.1 | Normative References | 5 |
| 1.2.2 | Informative References | 6 |
| 1.3 | Structure Overview | 6 |
| 1.4 | Relationship to Protocols and Other Structures | 6 |
| 1.5 | Applicability Statement | 7 |
| 1.6 | Versioning and Localization | 7 |
| 1.7 | Vendor-Extensible Fields | 7 |
| 2 | Structures..... | 8 |
| 2.1 | Log Data Fields | 8 |
| 2.1.1 | audiocodec | 8 |
| 2.1.2 | avgbandwidth | 9 |
| 2.1.3 | c-buffercount | 9 |
| 2.1.4 | c-cpu | 9 |
| 2.1.5 | c-dns | 10 |
| 2.1.6 | c-hostexe | 10 |
| 2.1.7 | c-hostexever..... | 10 |
| 2.1.8 | c-ip | 10 |
| 2.1.9 | c-max-bandwidth | 11 |
| 2.1.10 | c-os | 11 |
| 2.1.11 | c-osversion | 11 |
| 2.1.12 | c-pkts-lost-client | 12 |
| 2.1.13 | c-pkts-lost-cont-net | 12 |
| 2.1.14 | c-pkts-lost-net | 13 |
| 2.1.15 | c-pkts-received | 13 |
| 2.1.16 | c-pkts-recovered-ECC | 13 |
| 2.1.17 | c-pkts-recovered-resent | 14 |
| 2.1.18 | c-playerid | 14 |
| 2.1.19 | c-playerlanguage..... | 15 |
| 2.1.20 | c-playerversion | 15 |
| 2.1.21 | c-quality..... | 15 |
| 2.1.22 | c-rate..... | 16 |
| 2.1.23 | c-resendreqs..... | 16 |
| 2.1.24 | c-starttime | 17 |
| 2.1.25 | c-status..... | 17 |
| 2.1.25.1 | Status Code 200 (No Error) | 17 |
| 2.1.25.2 | Status Code 210 (Client Successfully Reconnected) | 17 |
| 2.1.26 | c-totalbuffertime | 17 |
| 2.1.27 | c-channelURL..... | 18 |
| 2.1.28 | c-bytes..... | 18 |
| 2.1.29 | cs-media-name | 18 |
| 2.1.30 | cs-media-role..... | 19 |
| 2.1.31 | cs-Referer | 20 |
| 2.1.32 | cs-url | 20 |
| 2.1.33 | cs-uri-stem | 20 |
| 2.1.34 | cs-User-Agent | 21 |
| 2.1.35 | cs-user-name..... | 22 |
| 2.1.36 | date..... | 22 |
| 2.1.37 | filelength | 22 |

| | | |
|----------|--|-----------|
| 2.1.38 | filesize | 22 |
| 2.1.39 | protocol..... | 23 |
| 2.1.40 | s-content-path | 23 |
| 2.1.41 | s-cpu-util..... | 23 |
| 2.1.42 | s-dns | 24 |
| 2.1.43 | s-ip | 24 |
| 2.1.44 | s-pkts-sent..... | 25 |
| 2.1.45 | s-proxied..... | 25 |
| 2.1.46 | s-session-id | 25 |
| 2.1.47 | s-totalclients | 25 |
| 2.1.48 | sc-bytes | 26 |
| 2.1.49 | time..... | 26 |
| 2.1.50 | transport | 26 |
| 2.1.51 | videocodec | 27 |
| 2.1.52 | x-duration | 27 |
| 2.2 | Logging Message: W3C Syntax | 27 |
| 2.2.1 | Basic Logging Syntax | 28 |
| 2.2.2 | Extended Logging Syntax | 28 |
| 2.2.3 | Connect-Time Logging Syntax..... | 29 |
| 2.3 | Logging Messages Sent to Web Servers | 29 |
| 2.4 | Logging Message: XML Schema | 30 |
| 2.5 | Legacy Log | 32 |
| 2.5.1 | Common Definitions..... | 32 |
| 2.5.2 | Legacy Log in W3C Format | 34 |
| 2.5.3 | Legacy Log in XML Format | 34 |
| 2.5.4 | Legacy Log Sent to a Web Server | 34 |
| 2.6 | Streaming Log | 34 |
| 2.6.1 | Common Definitions..... | 35 |
| 2.6.2 | Streaming Log Sent to Windows Media Services | 36 |
| 2.6.3 | Streaming Log Sent to a Web Server | 36 |
| 2.7 | Rendering Log | 37 |
| 2.7.1 | Common Definitions..... | 37 |
| 2.7.2 | Rendering Log Sent to Windows Media Services | 38 |
| 2.7.3 | Rendering Log Sent to a Web Server..... | 38 |
| 2.8 | Connect-Time Log | 39 |
| 3 | Structure Examples | 40 |
| 3.1 | Legacy Logging Message | 40 |
| 3.2 | Defining Custom Namespaces in an XML Log | 41 |
| 3.3 | Example Streaming Log Messages..... | 42 |
| 3.4 | Example Rendering Log Messages | 44 |
| 3.5 | Example Connect-Time Log Message | 45 |
| 3.6 | Example Log Sent to a Web Server | 46 |
| 3.7 | Parsing Windows Media Log Files..... | 46 |
| 4 | Security Considerations | 47 |
| 5 | Appendix A: Windows Behavior | 48 |
| 6 | Index..... | 49 |

1 Introduction

This specification defines the Windows Media Log Data Structure, a Microsoft proprietary interface. The Windows Media Log Data Structure is a syntax for logging messages. The logging messages specify information about how a client received multimedia content from a streaming server. For example, logging messages can specify how many packets were received and how long it took for the client to receive the content.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

Advanced Systems Format (ASF)
Globally Unique Identifier (GUID)

The following terms are defined in [\[MS-WMSP\]](#):

Content
Playlist
Session
Stream
Streaming

The following terms are specific to this document:

Client: The entity that has created the logging message, or an entity that receives a logging message from a client. In the latter case, the client is a proxy.

Proxy: An entity that can receive logging messages from both a client and a proxy, and/or a server that is streaming on behalf of another server.

Server: An entity that transfers content to a client through streaming. A server might be able to do streaming on behalf of another server, thus a server can also be a proxy.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[ASF] Microsoft Corporation, "Advanced Systems Format Specification", December 2004, http://download.microsoft.com/download/7/9/0/790fecaa-f64a-4a5e-a430-0bccdab3f1b4/ASF_Specification.doc

If you have any trouble finding [ASF], please check [here](#).

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, <http://www.ietf.org/rfc/rfc1945.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2326] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998, <http://www.ietf.org/rfc/rfc2326.txt>

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>

[RFC3066] Alvestrand, H., "Tags for the Identification of Language", RFC 3066, January 2001, <http://www.ietf.org/rfc/rfc3066.txt>

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", RFC 3629, November 2003, <http://www.ietf.org/rfc/rfc3629.txt>

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

[RFC4234] Crocker, D., Ed. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, <http://www.ietf.org/rfc/rfc4234.txt>

1.2.2 Informative References

[MS-MMSP] Microsoft Corporation, "[Microsoft Media Server \(MMS\) Protocol Specification](#)", June 2007.

[MS-MSB] Microsoft Corporation, "[Media Stream Broadcast \(MSB\) Protocol Specification](#)", January 2007.

[MS-RTSP] Microsoft Corporation, "[Real-Time Streaming Protocol \(RTSP\) Windows Media Extensions](#)", July 2007.

[MS-WMSP] Microsoft Corporation, "[Windows Media HTTP Streaming Protocol Specification](#)", March 2007.

[MSDN-WMMETA] Microsoft Corporation, "Windows Media Metafiles", <http://msdn2.microsoft.com/en-us/library/bb248407.aspx>

[MSFT-LOGPARSER] Microsoft Corporation, "Log Parser 2.2", <http://www.microsoft.com/downloads/details.aspx?FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07&displaylang=en>

[W3C-EXLOG] World Wide Web Consortium, "Extended Log File Format", <http://www.w3.org/TR/WD-logfile.html>

1.3 Structure Overview

The Windows Media Log Data Structure is a syntax for logging messages. The logging messages specify information about how a **client** received multimedia **content** from a **streaming server**.

1.4 Relationship to Protocols and Other Structures

The logging messages defined in this specification are used by the [Windows Media HTTP Streaming Protocol](#) and the [Real-Time Streaming Protocol \(RTSP\) Windows Media Extensions](#). When those two

protocols are used, the logging messages defined by this specification can be encapsulated in protocol messages specific to the streaming protocol in use. The resulting protocol messages are sent to either Windows Media Services or to a proxy compatible with the logging message syntax defined in this specification.

It is also possible to send logging messages to an HTTP Web server. This is possible when using the two streaming protocols mentioned above and when using two other streaming protocols: [Microsoft Media Server \(MMS\) Protocol](#) and [Media Stream Broadcast \(MSB\) Protocol](#).

1.5 Applicability Statement

The syntax for logging messages defined by this specification is applicable to implementations of the four streaming protocols mentioned in section [1.4](#).

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

Logging messages in XML format are vendor-extensible. Any logging information added by a vendor MUST be encoded using the "client-logging-data" syntax element specified in section [2.4](#).

2 Structures

Section [2.1](#) defines fields that can appear in a logging message. Not all fields appear in all logging messages, however. Section [2.2](#) defines the syntax of W3C-based logging messages, and section [2.4](#) defines the syntax of XML-based logging messages.

Section [2.5](#) defines the legacy logging message type. Section [2.6](#) defines the streaming log message type. Section [2.7](#) defines the rendering log message type. Section [2.8](#) defines the connection log message type.

The information contained in a logging message is always specific to a particular session. The extent of a session is defined by the streaming protocol used by the server. A rendering log message (as specified in section [2.7](#)) can be sent without streaming from a server, and, in that case, a session starts when the playback of the playlist starts, and stops when the playback of the playlist stops.

Below are some common Augmented Backus-Naur Form (ABNF) constructions, as specified in [\[RFC4234\]](#), that are used throughout this specification. Any ABNF syntax rules that are not defined in [\[RFC4234\]](#) or in this specification may be defined in [\[RFC1945\]](#) or [\[RFC2616\]](#).

```
date-year    = 4DIGIT          ; "19xx" and "20xx" typical
date-month   = 2DIGIT          ; 01 through 12
date-day     = 2DIGIT          ; 01 through 31
time-hour    = 2DIGIT          ; 00 through 24
time-min     = 2DIGIT          ; 00 through 59
time-sec     = 2DIGIT          ; 00 through 59, 60 if leap second

ip_addr      = IPv4address | IPv6address
              ; Defined in Appendix A of RFC3986

ver_major    = 1*2DIGIT
ver_minor    = 1*2DIGIT [ "." 1*4DIGIT "." 1*4DIGIT ]
```

2.1 Log Data Fields

2.1.1 audiocodec

This field SHOULD specify a list of audio codecs used to decode the audio **streams** accessed by the client. Each codec MUST be listed only once regardless of the number of streams decoded by that codec.

The value for **audiocodec** MUST NOT exceed 256 characters in total length. If the codec name is not available, then the field MUST be set to "-".

The syntax of the **audiocodec** field is defined as follows:

```
codec-name= 1*255VCHAR
audiocodec= "-" | ( codec-name *( ";" codec-name ) )
```

Example:

2.1.2 avgbandwidth

This field MUST specify the average bandwidth, in bits per second, at which the client received content from the server (which may be a **proxy**), as measured by the client from the start of the current session. This is only applicable during periods in which the server is streaming the content. Depending on the streaming protocol used, it might be possible for the session to be in a "paused" state in which streaming is suspended. The value for **avgbandwidth** does not account for the average bandwidth during such periods in which streaming is suspended.

If the notion of an average bandwidth is not applicable, for example, because the client did not receive any content from the server, then the field MUST be set to "-".

If the numerical value is specified, it MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **avgbandwidth** field is defined as follows:

```
avgbandwidth= "-" | 1*10DIGIT
```

Example:

```
102585
```

2.1.3 c-buffercount

This field MUST specify the number of times the client buffered while playing the content, counted from when the client most recently started streaming the content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-buffercount** field is defined as follows:

```
c-buffercount= 1*10DIGIT
```

Example:

```
1
```

2.1.4 c-cpu

This field MUST specify the type of CPU used by the client computer.

The syntax of the **c-cpu** field is defined as follows:

```
c-cpu= 1*64VCHAR
```

Example:

Pentium

2.1.5 c-dns

This field SHOULD be set to "-". The field MAY specify the DNS name of the client sending the log. [<1>](#)

The syntax of the **c-dns** field is defined as follows:

```
c-dns= "-"  
      | reg-name ; as defined in [RFC3986]
```

Example:

```
wmt.test.com
```

2.1.6 c-hostexe

This field specifies the file name of the host application executed on the client. This field MUST NOT refer to a .dll, .ocx, or other non-executable file.

The syntax of the **c-hostexe** field is defined as follows:

```
c-hostexe= *255VCHAR
```

Example:

```
wmplayer.exe
```

2.1.7 c-hostexever

This field MUST specify the version number of the host application running on the client.

The syntax of the **c-hostexever** field is defined as follows:

```
c-hostexever= ver_major "." ver_minor
```

Example:

```
6.2.5.323
```

2.1.8 c-ip

When a client creates a logging message, it SHOULD specify the **c-ip** field as "-" but MAY specify the IP address of the client.

If a proxy is forwarding a logging message on behalf of a client, the **c-ip** field MUST specify the IP address of the client. The proxy MUST replace the value of the **c-ip** field that was specified by the client with the IP address of the client (as known to the proxy).

The syntax of the **c-ip** field is defined as follows:

```
c-ip = "-" | ip_addr
```

Example:

```
157.100.200.300
```

Example:

```
3ffe:2900:d005:f28b:0000:5efe:157.55.145.142
```

2.1.9 c-max-bandwidth

This field MUST be set to "-".

The syntax of the **c-max-bandwidth** field is defined as follows:

```
c-max-bandwidth = "-"
```

Example:

```
-
```

2.1.10 c-os

This field MUST specify the client's operating system. [<2>](#)

The syntax of the **c-os** field is defined as follows:

```
OSname= "Windows_98" | "Windows_ME" | "Windows_NT"  
        | "Windows_2000" | "Windows_XP" | "Windows"  
        | "Windows_Server 2003"  
c-os = OSname | 1*64VCHAR
```

Example:

```
Windows
```

2.1.11 c-osversion

This field MUST specify the version number of the client's operating system.

The syntax of the **c-osversion** field is defined as follows:

```
c-osversion= ver_major "." ver_minor
```

Example:

```
6.0.0.6000
```

2.1.12 c-pkts-lost-client

This field MUST specify the number of ASF data packets ([\[ASF\]](#) section 5.2) lost during transmission from server to client and not recovered at the client layer through error correction or at the network layer by using the User Datagram Protocol (UDP) resends, counted from when the client most recently started streaming the content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-client** field is defined as follows:

```
c-pkts-lost-client= 1*10DIGIT
```

Example:

```
5
```

2.1.13 c-pkts-lost-cont-net

This field MUST specify the largest number of ASF data packets that were lost as a consecutive span during transmission from server to client and counted from when the client most recently started streaming the content.

For example, if data packets numbered 1, 4, and 8 are received, and packets 2, 3, 5, 6 and 7 are lost, then packets 2 and 3 constitute a span of two lost packets, and packets 5, 6 and 7 constitute a span of three lost packets. In this example, the **c-pkts-lost-cont-net** field would be set to 3—the size of the largest span.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-cont-net** field is defined as follows:

```
c-pkts-lost-cont-net= 1*10DIGIT
```

Example:

```
2
```

2.1.14 c-pkts-lost-net

This field MUST specify the number of ASF data packets lost on the network layer, counted from when the client most recently started streaming the content. Packets lost at the network layer can be recovered if the client re-creates them by using forward error correction.

The numerical difference between the value of **c-pkts-lost-net** and the value of [c-pkts-lost-client](#) MUST be equal to the value of [c-pkts-recovered-ECC](#).

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-lost-net** field is defined as follows:

```
c-pkts-lost-net= 1*10DIGIT
```

Example:

```
2
```

2.1.15 c-pkts-received

This field MUST specify the number of ASF data packets that have been correctly received by the client on the first attempt counted from when the client most recently started streaming the content. (ASF data packets that were received through error correction code (ECC) recovery or UDP resends are not included in the **c-pkts-received** field.)

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-received** field is defined as follows:

```
c-pkts-received= 1*10DIGIT
```

Example:

```
523
```

2.1.16 c-pkts-recovered-ECC

This field MUST specify the number of ASF data packets that were lost at the network layer but were subsequently recovered, counted from when the client most recently started streaming the content. The value of this field MUST be equal to the numerical difference between the value of [c-pkts-lost-net](#) and the value of [c-pkts-lost-client](#).

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-recovered-ECC** field is defined as follows:

```
c-pkts-recovered-ECC= 1*10DIGIT
```

Example:

2.1.17 c-pkts-recovered-resent

This field MUST specify the number of ASF data packets that were either recovered because they were resent through UDP or because they were received out of order.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-pkts-recovered-resent** field is defined as follows:

```
c-pkts-recovered-resent= 1*10DIGIT
```

Example:

```
5
```

2.1.18 c-playerid

This field specifies a unique identifier for the client application that originated the request. The identifier MUST be a **GUID**. The GUID is expressed in registry format and is not enclosed in quotation marks, as shown by the ABNF syntax below.

If the client is configured to remain anonymous (that is, not send private information), the client MUST set the **c-playerid** field as indicated by the ABNF syntax for the `playid_priv` syntax element as shown in the code example below. Otherwise, **c-playerid** MUST use the syntax for `playid_pub` as shown in the code example below. The client MUST choose a value for `playid_pub` randomly, and the same value MUST be used for `playid_pub` in all logging messages created by the client application, regardless of which content is streamed.

Furthermore, multiple instances, or incarnations, of the client application MUST use the same value for the `playid_pub` syntax element. However, if the client application is shared by multiple users, and it is possible to determine a user identity or account name of the user launching the client application, then the value for `playid_pub` SHOULD be different for each user identity or account name. For example, multi-user operating systems typically have separate accounts with a distinct account name for each user, while cellular telephones do not.

If the client uses the `playid_priv` syntax element, then the client SHOULD choose the value for the `playid` syntax element randomly; however, the client MUST use the same `playid` value for all logging messages sent for the same session.

The syntax of the **c-playerid** field is defined as follows:

```
playid= 12HEXDIG
playid_pub = "{" 8HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-"
              4HEXDIG "-" 12HEXDIG "}"
playid_priv= "{3300AD50-2C39-46c0-AE0A-" playid "}"

c-playerid= playid_pub / playid_priv
```

Example:

```
{c579d042-cecc-11d1-bb31-00a0c9603954}
```

Example (client is anonymous):

```
{3300AD50-2C39-46c0-AE0A-70b64f321a80}
```

2.1.19 c-playerlanguage

This field MUST specify the language-country code of the client.

The syntax of the **c-playerlanguage** field is defined as follows:

```
c-playerlanguage= Language-Tag  
; see section 2.1 of [RFC3066]
```

Example:

```
en-US
```

2.1.20 c-playerversion

This field MUST specify the version number of the client.

The syntax of the **c-playerversion** field is defined as follows:

```
c-playerversion = ver_major "." ver_minor
```

Example:

```
7.0.1024
```

2.1.21 c-quality

This field MUST specify the percentage of packets that were received by the client, counted from when the client most recently started streaming the content.

If **cPacketsRendered** represents all packets received by the client including packets recovered by ECC and UDP resend such that:

```
cPacketsRendered = c-pkts-received + c-pkts-recovered-ECC + c-pkts-  
recovered-resent
```

then the value for the **c-quality** field MUST be calculated as:

$$\left[\frac{\text{cPacketsRendered}}{\text{cPacketsRendered} + \text{c-pkts-lost-client}} \right] * 100$$

If the denominator in the above equation evaluates to 0, **c-quality** MUST be specified as 100.

The syntax of the **c-quality** field is defined as follows:

`c-quality = 1*2DIGIT / "100"`

Example:

89

2.1.22 c-rate

This field MUST specify the rate of streaming or playback as a multiplier of the normal streaming or playback rate.

For example, a value of 1 specifies streaming or playback at the normal rate, while a value of -5 indicates rewind at a speed 5 times faster than real-time, and a value of 5 indicates fast-forward at a rate 5 times faster than real-time.

For Legacy and Streaming Logs, **c-rate** MUST be the streaming rate. For Rendering logs, **c-rate** MUST be the rendering (playback) rate.

The value of **c-rate** MUST reflect the rate that was in effect at the beginning of the period covered by the logging message because streaming or playback might already have ended by the time the logging message is generated.

The syntax of the **c-rate** field is defined as follows:

`c-rate= ["-"] 1*2DIGIT`

Example:

1

2.1.23 c-resendreqs

This field MUST specify the number of requests made by the client to receive lost ASF data packets, counted from when the client most recently started streaming the content. If the client is not using UDP resend, the value of this field MUST be "-".

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-resendreqs** field is defined as follows:

`c-resendreqs= "-" / 1*10DIGIT`

Example:

2.1.24 c-starttime

This field MUST specify the time offset, in seconds, in the content from which the client started to render content. This represents the presentation time of the ASF data packets that the client began rendering. For live broadcasts, the client MUST set this field to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-starttime** field is defined as follows:

```
c-starttime= 1*10DIGIT
```

Example:

```
39
```

2.1.25 c-status

This field MUST specify a numerical code that indicates the status of the client that creates the logging message.

The syntax of the **c-status** field is defined as follows:

```
c-status= "200" / "210"
```

Example:

```
200
```

2.1.25.1 Status Code 200 (No Error)

This code indicates that the client successfully streamed and submitted the log.

2.1.25.2 Status Code 210 (Client Successfully Reconnected)

This code indicates that the client disconnected and then reconnected to the server. [<3>](#)

2.1.26 c-totalbuffertime

This field MUST specify the total time, in seconds, that the client spent buffering the ASF data packets in the content, counted from when the client most recently started streaming the content. If the client buffers the content more than once before a log is generated, **c-totalbuffertime** MUST be equal to the total amount of time that the client spent buffering the ASF data packets.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-totalbuffertime** field is defined as follows:

```
c-totalbuffertime= 1*10DIGIT
```

Example:

```
20
```

2.1.27 c-channelURL

This field MUST specify the URL to the multicast station (.nsc) file (for more information, see [\[MS-MSB\]](#)) if such a file was used by the client. Whenever an .nsc file is used, this field MUST be specified, even if the MSB Protocol was not used to stream content.

The syntax of the **c-channelURL** field is defined as follows:

```
c-channelURL = "-"  
              / URI-reference ; as defined in section 4.1 of [RFC3986].
```

Example:

```
http://server/channel.nsc
```

2.1.28 c-bytes

This field MUST specify the number of bytes received by the client from the server, counted from when the client most recently started streaming the content.

The value for the **c-bytes** field MUST NOT include TCP/IP or other overhead added by the network stack. Higher-level protocols such as HTTP [\[RFC2616\]](#), RTSP [\[RFC2326\]](#), and the MMS Protocol [\[MS-MMSP\]](#), can each introduce differing amounts of overhead, resulting in different values for the same content.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **c-bytes** field is defined as follows:

```
c-bytes= 1*10DIGIT
```

Example:

```
28000
```

2.1.29 cs-media-name

The purpose of this field is to specify the file name of the content or server-side **playlist** entry that was streamed or played by the client. For Legacy and Streaming Logs, the value of this field MUST be the content or server-side playlist entry that was streamed. For Rendering Logs, it MUST be the content or server-side playlist entry that was rendered (played).

If the server provided a Content Description, (see, for example, the [Windows Media HTTP Streaming Protocol](#)), and the Content Description contains an entry named WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL, the value of the **cs-media-name** field MUST be equal to the value of the WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL entry.

Otherwise, if the client is using an Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)), and the file specifies a logging parameter called "cs-media-name", then the value of the **cs-media-name** field in the logging message MUST be equal to the value of the "cs-media-name" logging parameter in the .asx file. See section [3.2](#) for an example of how this parameter is specified in an .asx file.

If none of the above applies, **cs-media-name** MUST be specified as "-".

The syntax of the **cs-media-name** field is defined as follows:

```
cs-media-name= *VCHAR
```

Examples:

```
C:\wmpub\wmroot\MyAd2.asf
```

2.1.30 cs-media-role

The purpose of this field is to specify a value that can be associated with a server-side playlist entry to signify the role of the playlist entry. For Legacy and Streaming logs, the value of this field MUST be the role of the server-side playlist entry that was streamed. For Rendering Logs, it MUST be the role of the server-side playlist entry that was rendered (played).

If the server provided a Content Description, (see, for example, the [Windows Media HTTP Streaming Protocol](#)), and the Content Description contains an entry named WMS_CONTENT_DESCRIPTION_ROLE, the value of the **cs-media-role** field MUST be equal to the value of the WMS_CONTENT_DESCRIPTION_ROLE entry.

Otherwise, if the client is using an Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)), and the file specifies a logging parameter called "cs-media-role", then the value of the **cs-media-role** field in the logging message MUST be equal to the value of the "cs-media-role" logging parameter in the .asx file. See section [3.2](#) for an example of how this parameter is specified in an .asx file.

If none of the above applies, the **cs-media-role** MUST be specified as "-".

The syntax of the **cs-media-role** field is defined as follows:

```
cs-media-role= *VCHAR
```

Example:

```
ADVERTISEMENT
```

2.1.31 cs-Referer

This field SHOULD specify the URL to the Web page that the client software application is embedded within, except if the client software application was not embedded in a Web page. If the client software application is not embedded in a Web page, but an Active Stream Redirector (ASX) file (for more information, see [\[MSDN-WMMETA\]](#)) was obtained from a Web page, then this field SHOULD be set to the URL to that Web page.

If none of the above applies, this field MUST be set to "-".

The syntax of the **cs-Referer** field is defined as follows:

```
cs-Referer= "-"  
           / URI-reference ; as defined in section 4.1 of [RFC3986]
```

Examples:

```
http://www.adventure-works.com/default.htm
```

2.1.32 cs-url

This field MUST specify the URL for the streaming content originally requested by the client.

Note that the value of this field can be different from the URL actually used if the server redirected the client to a different URL, or if the client decided to use a streaming protocol that is different from the one indicated by the URL scheme of the original URL.

When the [MSB Protocol](#) is used, the "asfm" MUST be used as the URL scheme in the **cs-url** field.

The syntax of the **cs-url** field is defined as follows:

```
cs-url= URI-reference; as defined in section 4.1 of [RFC3986].
```

Example 1:

```
mms://www.adventure-works.com/some/content.asf
```

Example 2:

```
asfm://239.1.2.3:9000
```

2.1.33 cs-uri-stem

This field MUST specify the URL actually used by the client. Any query strings MUST be excluded from the URL. (This means that the value of the **cs-uri-stem** field is equal to the URL actually used by the client, truncated at the first "?" character.)

Note that the value of this field can be different from the URL originally requested by the client if the server redirected the client to a different URL, or if the client decided to use a streaming protocol that is different from the one indicated by the URL scheme of the original URL.

When the [Media Stream Broadcast \(MSB\) Protocol](#) is used (for more information, see [MS-MSB]), the "asfm" MUST be used as the URL scheme in the **cs-uri-stem** field.

The syntax of the **cs-uri-stem** field is defined as follows:

```
cs-uri-stem= URI-reference; as defined in section 4.1 of [RFC3986].
```

Example:

```
rtspt://server/test/sample.asf
```

2.1.34 cs-User-Agent

The purpose of this field is to specify information regarding the client application that is sending the logging message.

The **cs-User-Agent** field SHOULD be set to the same value that Internet Explorer specifies on the User-Agent HTTP protocol header. The field MAY be set differently as long as it adheres to the ABNF syntax as shown in the code example below.

If a logging message is forwarded by a proxy, the **cs-User-Agent** field MUST begin with the string "_via_". The original value specified by the client (which may be another proxy) on the **cs-User-Agent** field SHOULD be discarded. The proxy SHOULD include a product token on the **cs-User-Agent** field that specifies the brand and version of the proxy.

The syntax of the **cs-User-Agent** field is defined as follows:

```
cs-User-Agent= [ "_via_HTTP/1.0_" ]  
1*( product; [RFC2616] section 3.8  
| comment ); [RFC2616] section 2.2
```

Example 1: media player embedded in Internet Explorer 6 on Windows XP SP2:

```
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1;_SV1)
```

Example 2: application based on Windows Media Format 9 Series SDK:

```
Application/2.3 (WMFSDK/9.0.1234)
```

Example 3: proxy:

```
_via_HTTP/1.0_WMCaCheProxy/9.00.00.1234
```

2.1.35 cs-user-name

This field MUST be set to "-".

The syntax of the **cs-user-name** field is defined as follows:

```
cs-user-name= "-"
```

Example:

```
-
```

2.1.36 date

This field MUST specify the current date on the client when the log message is created. The time MUST be specified in UTC.

The syntax of the **date** field is defined as follows:

```
date= date-year "-" date-month "-" date-day
```

Example:

```
1997-10-09
```

2.1.37 filelength

This field MUST specify the length of the **ASF** file, in seconds. For a live broadcast stream, the value for **filelength** is undefined and MUST be set to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **filelength** field is defined as follows:

```
filelength= 1*10DIGIT
```

Example:

```
60
```

2.1.38 filesize

This field MUST specify the size of the ASF file, in bytes. For a live broadcast stream, the value for the **filesize** field is undefined and MUST be set to zero.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **filesize** field is defined as follows:

```
filesize= 1*10DIGIT
```

Example:

```
86000
```

2.1.39 protocol

This field MUST specify the protocol used to stream content to the client.

If the [Windows Media HTTP Streaming Media Protocol](#) was used, the value of the **protocol** field MUST be "http".

If the [RTSP Windows Media Extensions](#) was used, and all ASF data packets were transmitted over TCP, the value of the **protocol** field MUST be "rtspt". If some ASF data packets were transmitted over UDP, the value of the **protocol** field MUST be "rtspu".

If the [MSB Protocol](#) was used, the value of the **protocol** field MUST be "asfm".

Note The value for **protocol** can be different from the URL moniker used in the stream request.

The syntax of the **protocol** field is defined as follows:

```
protocol= "http" / "rtspt" / "rtspu" / "asfm"
```

Example:

```
http
```

2.1.40 s-content-path

This field MUST be set to "-".

The syntax of the **s-content-path** field is defined as follows:

```
s-content-path = "-"
```

Example:

```
-
```

2.1.41 s-cpu-util

When a client creates a logging message, it MUST specify the **s-cpu-util** field as "-".

If a proxy is forwarding the logging message on behalf of a client (which may be another proxy), the proxy MUST replace the value of the **s-cpu-util** field that was specified by the client with the proxy's current CPU load, in percentage, at the time of forwarding the logging message. If the proxy uses

symmetric multi-processing, the CPU load value MUST be calculated as the average for all processors.

When a numerical value is specified, the value MUST be an integer in the range from 0 through 100.

The syntax of the **s-cpu-util** field is defined as follows:

```
s-cpu-util = "-" | 1*2DIGIT | "100"
```

Example:

```
40
```

2.1.42 s-dns

This field SHOULD specify the DNS name of the proxy if a proxy is forwarding the logging message on behalf of a client (which may be another proxy). The proxy MUST replace the value of the **s-dns** field that was specified by the client with its own DNS name or with "-" if the DNS name cannot be determined.

When a client creates a logging message, it SHOULD specify the **s-dns** field as "-" but MAY specify the DNS name of the server that the clientstreamed the content from.

The syntax of the **s-dns** field is defined as follows:

```
s-dns= "-"  
      | reg-name ; as defined in [RFC3986].
```

Example:

```
wmt.adventure-works.com
```

2.1.43 s-ip

For Legacy and Streaming Logs, this field MUST specify the IP address of the server that the client streamed the content from.

For Rendering Logs, the field MUST specify the IP address of the proxy if a proxy is forwarding the logging message on behalf of a client. The proxy MUST replace the value of the **s-ip** field that was specified by the client (which may be another proxy) with the IP address used by the proxy when forwarding the Rendering Log to the server (which may be another proxy).

When a client creates a rendering log, it SHOULD specify the **s-ip** field as "-" but can specify the IP address of the server that the clientstreamed the content from.

The syntax of the **s-ip** field is defined as follows:

```
s-ip = "-" | ip_addr
```

Example:

155.12.1.234

2.1.44 s-pkts-sent

This field MUST be set to "-".

The syntax of the **s-pkts-sent** field is defined as follows:

```
s-pkts-sent= "-"
```

Example:

-

2.1.45 s-proxied

This field MUST be set to "1" in a logging message that is being forwarded by a proxy. The client that creates the logging message MUST set the field to "0" and the proxy MUST change the value to "1" when it forwards the logging message.

The syntax of the **s-proxied** field is defined as follows:

```
s-proxied= "0" / "1"
```

Example:

1

2.1.46 s-session-id

This field MUST be set to "-".

The syntax of the **s-session-id** field is defined as follows:

```
s-session-id= "-"
```

Example:

-

2.1.47 s-totalclients

When a client creates a logging message, it MUST specify the **s-totalclients** field as "-".

If a proxy is forwarding the logging message on behalf of a client (which may be another proxy), then the proxy MUST replace the value of the **s-totalclients** field that was specified by the client with the total number of clients connected to the proxy server (for all target servers combined).

When a numerical value is specified, the value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **s-totalclients** field is defined as follows:

```
s-totalclients = "-" | 1*10DIGIT
```

Example:

```
201
```

2.1.48 sc-bytes

This field MUST be set to "-".

The syntax of the **sc-bytes** field is defined as follows:

```
sc-bytes= "-"
```

Example:

```
-
```

2.1.49 time

This field MUST specify the current time on the client when the log message is created. The time MUST be specified in UTC.

The syntax of the **time** field is defined as follows:

```
time= time-hour ":" time-min ":" time-sec
```

Example:

```
15:30:30
```

2.1.50 transport

This field MUST specify the transport protocol used to receive the ASF data packets.

The syntax of the **transport** field is defined as follows:

```
transport= "UDP" | "TCP"
```

Example:

```
UDP
```

2.1.51 videocodec

This field SHOULD specify a list of video codecs that are used to decode the video streams accessed by the client. Each codec MUST be listed only once, regardless of the number of streams decoded by that codec.

The value for **videocodec** MUST NOT exceed 256 characters in total length. If the codec name is not available, then the field MUST be set to "-".

The syntax of the **videocodec** field is defined as follows:

```
codec-name= 1*255VCHAR
videocodec= "-" | ( codec-name *( ";" codec-name ) )
```

Example:

```
Microsoft_MPEG-4_Video_Codec_V2
```

2.1.52 x-duration

For Legacy and Rendering Log messages, this field MUST specify how much of the content has been rendered (played) to the end user, specified in seconds. Time spent buffering data MUST NOT be included in this value.

Playback at non-normal play speed does not affect the amount of content rendered, when expressed in time units. For example, if the client was rewinding the content, the **x-duration** value can be computed as the absolute value of the difference between the starting presentation time and ending presentation time.

For Streaming Log messages, the **x-duration** field MUST specify the time it took to receive the content, in seconds.

Fractional time amounts MUST be rounded to the nearest larger integer value.

The value MUST be an integer in the range from 0 through 4,294,967,295.

The syntax of the **x-duration** field is defined as follows:

```
x-duration= 1*10DIGIT
```

Example:

```
31
```

2.2 Logging Message: W3C Syntax

A World Wide Web Consortium (W3C) format logging message consists of the values of various fields, each value separated from the next by a single space character. Logging messages that adhere to this syntax are said to use the W3C format because the syntax is conformant with the syntax for logging entries in the Extended Log File Format (for more information, see [\[W3C-EXLOG\]](#)), which is defined by W3C.

Section [2.2.1](#) specifies the W3C format syntax used in most logging messages. Section [2.2.2](#) specifies the W3C format syntax used in certain Rendering log messages, and section [2.2.3](#) specifies the W3C format syntax used in Connect-time log messages.

The sections mentioned above define the ordering of the fields in the W3C format syntax but not how the values of the fields are assigned. The rules governing the values of the individual fields depend on the logging message in which the W3C format syntax is used. For example, the [s-ip](#) field is used as defined in section [2.1.43](#) for some logging messages, while other logging messages provide an alternate definition of the **s-ip** field.

All W3C format syntax MUST use the UTF-8 character set as specified in [\[RFC3629\]](#). In any fields that specify a URL, such as [cs-uri](#), the URL MUST be encoded using percent-encoding, as specified in [\[RFC3986\]](#) section 2.1.

A single dash character (which is represented by U+002D and by "-" in ABNF syntax) MUST be used to indicate that the value is empty — that is, it is either not available or not applicable.

All spaces embedded within a field value MUST be replaced by an underscore character (which is represented by U+005F and by "_" in ABNF syntax). For example, "MPEG Layer-3" would be transformed into "MPEG_Layer-3" in a W3C-format logging message.

Note Transformations defined in this section are not necessarily reversible. Methods for parsing, analyzing, or extracting information from logging messages are implementation-specific and are outside the scope of this specification.

2.2.1 Basic Logging Syntax

Most logging messages contain logging information in W3C format, adhering to the syntax specified below. The logging information consists of either 44 or 47 fields.

```
log_data44 = c-ip SP date SP time SP c-dns SP cs-uri-stem SP c-starttime SP
            x-duration SP c-rate SP c-status SP c-playerid SP
            c-playerversion SP c-playerlanguage SP cs-User-Agent SP
            cs-Referer SP c-hostexe SP c-hostexever SP c-os SP c-osversion SP
            c-cpu SP filelength SP filesize SP avgbandwidth SP protocol SP
            transport SP audiocodec SP videocodec SP c-channelURL SP sc-bytes SP
            c-bytes SP s-pkts-sent SP c-pkts-received SP c-pkts-lost-client SP
            c-pkts-lost-net SP c-pkts-lost-cont-net SP c-resendreqs SP
            c-pkts-recovered-ECC SP c-pkts-recovered-resent SP c-buffercount SP
            c-totalbuffertime SP c-quality SP s-ip SP s-dns SP
            s-totalclients SP s-cpu-util
            [ SP cs-url SP cs-media-name SP cs-media-role ]
```

2.2.2 Extended Logging Syntax

Certain types of "rendering" log messages (section [2.7](#)) contain logging information in the W3C format defined below. This logging information consists of 52 fields:

```
log_data52 = c-ip SP date SP time SP c-dns SP cs-uri-stem SP c-starttime SP
            x-duration SP c-rate SP c-status SP c-playerid SP
            c-playerversion SP c-playerlanguage SP cs-User-Agent SP
            cs-Referer SP c-hostexe SP c-hostexever SP c-os SP c-osversion SP
            c-cpu SP filelength SP filesize SP avgbandwidth SP protocol SP
            transport SP audiocodec SP videocodec SP c-channelURL SP sc-bytes SP
```

```
c-bytes SP s-pkts-sent SP c-pkts-received SP c-pkts-lost-client SP
c-pkts-lost-net SP c-pkts-lost-cont-net SP c-resendreqs SP
c-pkts-recovered-ECC SP c-pkts-recovered-resent SP c-buffercount SP
c-totalbuffertime SP c-quality SP s-ip SP s-dns SP
s-totalclients SP s-cpu-util SP cs-user-name SP s-session-id SP
s-content-path SP cs-url SP cs-media-name SP c-max-bandwidth SP
cs-media-role SP s-proxied
```

2.2.3 Connect-Time Logging Syntax

Connect-time log messages (section [2.8](#)) contain logging information in the W3C format defined below. This logging information consists of eight fields.

```
log_data8 = c-dns SP c-ip SP c-os SP c-osversion SPdate SP time SP
           c-cpu SP transport
```

2.3 Logging Messages Sent to Web Servers

Most of the logging messages defined in this specification can be sent to a HTTP Web server. The URL for the HTTP Web server for which logging messages are submitted can be specified in an Active Stream Redirector (ASX) file (for more information, see [\[MSDN-WMMETA\]](#)). Some of the compatible streaming protocols (listed in section [1.4](#)) can also specify the HTTP Web server URL through mechanisms that are specific to the streaming protocol. The syntax for the logging URL is defined as follows:

```
log-URL = Request-URI
```

The resource that is identified by log-URL MUST be capable of accepting and responding to the HTTP GET and POST request methods described in this section; however, the methods for doing so are implementation-specific.

Prior to sending a logging message to a Web server, a client SHOULD send an HTTP GET request to the specified Web server URL to validate the URL. The logging validation request MUST adhere to the following ABNF syntax:

```
web-server-validate = "GET" SP log-URL SP HTTP-Version CRLF
                    *( VCHAR /CLRF )
```

The web server's response MUST adhere to the following ABNF syntax:

```
web-server-validate-response = HTTP-Version "200 OK" CRLF
                              *( VCHAR / CRLF ) "<body><h1>"
                              ( "NetShow ISAPI Log Dll" /
                                ( "WMS ISAPI Log Dll/"
                                  1*4DIGIT "." 1*4DIGIT "." 1*4DIGIT "." 1*4DIGIT ) )
                              *( VCHAR / CRLF ) "</h1>" *( VCHAR / CRLF ) </body>" *( VCHAR / CRLF )
```

The client SHOULD send the logging message to the Web server if the server's response adheres to the syntax for web-server-validate-response, above. If the client sent a request to validate the URL, and the server's response does not adhere to the syntax for web-server-validate-response, then this might mean that the URL is invalid. In this case, the client SHOULD NOT send the logging message.

When sending the logging message, the client MUST include the logging message in the body of a HTTP POST request.

All logging message requests that are sent to a Web server MUST adhere to the following ABNF syntax:

```
web-server-request = "POST" SP log-URL SP HTTP-Version CRLF
                    *( VCHAR / CRLF )
                    web-server-log
```

The logging message sent in the web-server-request message body MUST adhere to the following ABNF syntax:

```
web-server-log = "MX_STATS_LogLine:" SP TAB
                log_data44; defined in section 2.2.1
```

All HTTP GET and POST requests sent by the client or Web server must be syntactically correct as per [RFC1945](#) or [RFC2616](#). Any header or content element not explicitly represented in one of the preceding ABNF syntax examples MUST be ignored by the recipient.

For an example of logging URL validation and the subsequent transmission of a logging message to a Web server, see section [3.6](#).

2.4 Logging Message: XML Schema

Logging messages can be represented in XML. This section defines the schema used by all logging messages for which an XML representation has been defined with the exception of the [Connect-Time Log](#). The XML scheme for the Connect-Time Log is defined in section [2.8](#).

The XML-format log embeds W3C-format logging information inside the "Summary" XML tag. Individual logging fields are also represented using their own XML tags.

If the entity that generates the XML-format logging message (that is, the client) has access to a Content Description, then each name/value pair in the Content Description SHOULD be encoded as shown by the "contentdescription" syntax element in the ABNF syntax as shown in the code example below.

The Content Description is a data structure that is provided by Windows Media Services. If no Content Description is available to the client, then the "contentdescription" syntax element MUST NOT be included in the XML-format logging message.

If the entity that generates the XML-format logging message (that is, the client) submits additional or custom logging information, then it SHOULD be encoded as shown by the "client-logging-data"

syntax element in the ABNF syntax below. For an example illustrating submission of custom logging information, see section [3.2](#).

If no additional logging information is available, the "client-logging-data" syntax element MUST NOT be included in the XML-format logging message.

The XML-format logging syntax is defined using ABNF as shown in the code example below. Although not explicitly shown by the syntax, linear white space, including CR LF sequences, is allowed on each side of XML tags.

```
xml-tag = 1*ALPHA
cd-name = xml-tag
cd-value = xml-tag
cd-name-value-pair = "<" cd-name ">"
cd-value
    "</" cd-name ">"

contentdescription = "<ContentDescription>"
    *cd-name-value-pair
    "</ContentDescription>"

client-logging-data = "<" xml-tag ">"
    *cdl-name-value-pair
    "</" xml-tag ">"

xml-log = "<XML>"
    "<Summary>" summary-log "</Summary>"
    "<c-ip>" "0.0.0.0" "</c-ip>"
    "<date>" date "</date>"
    "<time>" time "</time>"
    "<c-dns>" c-dns "</c-dns>"
    "<cs-uri-stem>" cs-uri-stem "</cs-uri-stem>"
    "<c-starttime>" c-starttime "</c-starttime>"
    "<x-duration>" x-duration "</x-duration>"
    "<c-rate>" c-rate "</c-rate>"
    "<c-status>" c-status "</c-status>"
    "<c-playerid>" c-playerid "<c-playerid>"
    "<c-playerversion>" c-playerversion "</c-playerversion>"
    "<c-playerlanguage>" c-playerlanguage "</c-playerlanguage>"
    "<cs-User-Agent>" cs-User-Agent "</cs-User-Agent>"
    "<cs-Referer>" cs-Referer "<cs-Referer>"
    "<c-hostexe>" c-hostexe "</c-hostexe>"
    "<c-hostexever>" c-hostexever "</c-hostexever>"
    "<c-os>" c-os "</c-os>"
    "<c-osversion>" c-osversion "</c-osversion>"
    "<c-cpu>" c-cpu "</c-cpu>"
    "<filelength>" filelength "</filelength>"
    "<filesize>" filesize "</filesize>"
    "<avgbandwidth>" avgbandwidth "</avgbandwidth>"
    "<protocol>" protocol "</protocol>"
    "<transport>" transport "</transport>"
    "<audiocodec>" audiocodec "</audiocodec>"
    "<videocodec>" videocodec "</videocodec>"
    "<c-channelURL>" c-channelURL "</c-channelURL>"
    "<sc-bytes>" sc-bytes "</sc-bytes>"
    "<c-bytes>" c-bytes "</c-bytes>"
    "<s-pkts-sent>" s-pkts-sent "</s-pkts-sent>"
    "<c-pkts-received>" c-pkts-received "</c-pkts-received>"
```



```

"<c-pkts-lost-client>" c-pkts-lost-client "</c-pkts-lost-client>"
"<c-pkts-lost-net>" c-pkts-lost-net "</c-pkts-lost-net>"
"<c-pkts-lost-cont-net>" c-pkts-lost-cont-net "</c-pkts-lost-cont-net>"
"<c-resendsreqs>" c-resendsreqs "</c-resendsreqs>"
"<c-pkts-recovered-ECC>" c-pkts-recovered-ECC "</c-pkts-recovered-ECC>"
"<c-pkts-recovered-resent>" c-pkts-recovered-resent "</c-pkts-recovered-resent>"
"<c-buffercount>" c-buffercount "</c-buffercount>"
"<c-totalbuffertime>" c-totalbuffertime "</c-totalbuffertime>"
"<c-quality>" c-quality "</c-quality>"
"<s-ip>" "-" "</s-ip>"
"<s-dns>" "-" "</s-dns>"
"<s-totalclients>" "-" "</s-totalclients>"
"<s-cpu-util>" "-" "</s-cpu-util>"
"<cs-url>" cs-url "</cs-url>"
[ contentdescription ]
*client-logging-data
"</XML>"

```

The syntax only defines the ordering of the fields and the XML tag assigned to each field; it does not define how the values of the fields are assigned. The rules governing the values of the individual fields depend on the logging message in which the XML-format syntax is used.

The XML-format logging syntax MUST use the UTF-8 character set, as specified in [RFC3629](#). In any fields that specify a URL, such as cs-url, the URL MUST be encoded using percent-encoding, as specified in [RFC3986](#) section 2.1.

A single dash character (which is represented by U+002D and by "-" in ABNF syntax) MUST be used to indicate that the value is empty — that is, it is either not available or not applicable.

All spaces embedded within a field value MUST be replaced by an underscore character (which is represented by U+005F and by "_" in ABNF syntax). For example, "MPEG Layer-3" would be transformed into "MPEG_Layer-3" in a W3C-format logging message.

2.5 Legacy Log

The Legacy Log is also called a combination log because it contains both rendering and streaming information. The Legacy Log can be either in W3C format or XML format. A Legacy Log can be sent either to Windows Media Services or to a Web server.

2.5.1 Common Definitions

The following ABNF syntax rules applies to all variants of the legacy log: [<4>](#)

```

s-cpu-util      = "-"
c-ip            = "0.0.0.0"
s-dns          = "-"

```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **audiocodec**
- **avgbandwidth**
- **c-buffercount**

- **c-channelURL**
- **c-cpu**
- **c-dns**
- **c-hostexe**
- **c-hostexever**
- **c-os**
- **c-osversion**
- **c-pkts-lost-client**
- **c-pkts-lost-cont-net**
- **c-pkts-lost-net**
- **c-pkts-recovered-ECC**
- **c-pkts-recovered-resent**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-quality**
- **c-rate**
- **c-resendreqs**
- **c-starttime**
- **c-status**
- **c-totalbuffertime**
- **cs-Referer**
- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **protocol**

- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **transport**
- **videocodec**
- **x-duration**

The [Legacy Log](#) SHOULD include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.[<5>](#)

2.5.2 Legacy Log in W3C Format

The ABNF syntax for a [Legacy Log](#) in W3C format that is sent to Windows Media Services is defined as follows:

```
legacy-log-W3C      = log_data44          ; defined in section 2.2.1
s-ip                = "-"
```

2.5.3 Legacy Log in XML Format

The ABNF syntax for a [Legacy Log](#) in XML format that is sent to Windows Media Services is defined as follows:[<6>](#)

```
legacy-log-XML     = xml-log            ; defined in section 2.4
summary-log       = log_data44        ; defined in section 2.2.1
s-ip              = "-"
```

2.5.4 Legacy Log Sent to a Web Server

The ABNF syntax for a [Legacy Log](#) that is submitted to a Web server is defined as follows:

```
legacy-web-server-log = web-server-log    ; defined in section 2.3
```

The value of the **s-ip** field MUST be assigned as defined in section [2.1.43](#).

2.6 Streaming Log

The Streaming Log specifies how the client received streaming data but not how the client rendered the data. A Streaming Log can be sent either to Windows Media Services or to a Web server.

2.6.1 Common Definitions

The following ABNF syntax rules applies to all variants of the [Streaming Log](#):

```
audiocodec      = "-"
c-ip            = "0.0.0.0"
s-cpu-util     = "-"
s-dns          = "-"
videocodec     = "-"
```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **avgbandwidth**
- **c-buffercount**
- **c-channelURL**
- **c-cpu**
- **c-dns**
- **c-hostexe**
- **c-hostexever**
- **c-os**
- **c-osversion**
- **c-pkts-lost-client**
- **c-pkts-lost-cont-net**
- **c-pkts-lost-net**
- **c-pkts-recovered-ECC**
- **c-pkts-recovered-resent**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-quality**
- **c-rate**
- **c-resendreqs**
- **c-starttime**
- **c-status**
- **c-totalbuffertime**

- **cs-Referer**
- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **protocol**
- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **transport**
- **x-duration**

The Streaming Log MUST include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.

2.6.2 Streaming Log Sent to Windows Media Services

The [Streaming Log](#) sent to Windows Media Services is in XML format and MUST adhere to the following ABNF syntax: [<7>](#)

```
streaming-log      = xml-log           ; defined in section 2.4
summary-log       = log_data44        ; defined in section 2.2.1
s-ip              = "-"
```

2.6.3 Streaming Log Sent to a Web Server

The ABNF syntax for a [Streaming Log](#) that is submitted to a Web server is defined as follows:

```
streaming-web-server-log = web-server-log; defined in section 2.3
```

The value of the **s-ip** field MUST be assigned as specified in section [2.1.43](#).

2.7 Rendering Log

The Rendering Log describes playback of content by a client and is submitted to the upstream origin server (or a configured proxy) when the client ends playback. A Rendering Log can be sent either to Windows Media Services or to a Web server.

2.7.1 Common Definitions

The following ABNF syntax rules apply to all variants of the [Rendering Log](#):

```
avgbandwidth           = "-"
c-buffercount          = "-"
c-pkts-lost-client     = "-"
c-pkts-lost-cont-net   = "-"
c-pkts-lost-net        = "-"
c-pkts-received        = "-"
c-pkts-recovered-ECC   = "-"
c-pkts-recovered-resent = "-"
c-quality              = "100"
c-resendreqs           = "-"
c-totalbuffertime     = "-"
protocol               = "Cache"
transport              = "-"
```

The values of the following fields MUST be assigned as defined in section [2.1](#) :

- **audiocodec**
- **c-channelURL**
- **c-cpu**
- **c-hostexe**
- **c-hostexever**
- **c-ip**
- **c-os**
- **c-osversion**
- **c-playerid**
- **c-playerlanguage**
- **c-playerversion**
- **c-rate**
- **c-starttime**
- **c-status**
- **cs-Referer**

- **cs-media-name**
- **cs-uri-stem**
- **cs-url**
- **cs-User-Agent**
- **date**
- **filelength**
- **filesize**
- **s-cpu-util**
- **s-dns**
- **cs-media-role**
- **s-pkts-sent**
- **s-totalclients**
- **sc-bytes**
- **time**
- **videocodec**
- **x-duration**

The Rendering Log MUST include the optional fields **cs-url**, **cs-media-name**, and **cs-media-role**.

2.7.2 Rendering Log Sent to Windows Media Services

The [Rendering Log](#) sent to Windows Media Services is in XML format and MUST adhere to the following ABNF syntax:

```
rendering-log      = xml-log           ; defined in section 2.4
summary-log       = log_data52       ; defined in section 2.2.2
```

The values of the following fields MUST be assigned as defined in section [2.1](#): **c-max-bandwidth**, **cs-user-name**, **s-content-path**, **s-ip**, **s-proxied**, and **s-session-id**.

2.7.3 Rendering Log Sent to a Web Server

The ABNF syntax for a [Rendering Log](#) that is submitted to a Web server is defined as follows:

```
rendering-web-server-log = web-server-log; defined in section 2.3
```

The value of the **c-ip** field MUST be assigned as defined in section [2.1.8](#). The value of the **s-ip** field MUST be assigned as defined in section [2.1.43](#).

2.8 Connect-Time Log

The purpose of the Connect-Time Log is to specify some minimal amount of logging information about the client. It can be useful in cases where a client starts to stream some content but is disconnected from the network before it has the opportunity to create a [Streaming Log](#).

If a client sends a Connect-Time Log to the server at the start of the streaming **session**, the Connect-Time Log ensures that the server has received at least this minimal logging information in the case where the client subsequently is disconnected from the network.

Connect-Time Logs are not defined for Web servers. Connect-Time Logs are only defined in XML format, and the ABNF syntax is as follows:

```
connect-time-log = "<XML>"
  "<Summary>"
    log_data8                               ; defined in section 2.2.3
  "</Summary>"
  "<c-dns>" c-dns "</c-dns>"
  "<c-ip>" c-ip "</c-ip>"
  "<c-os>" c-os "</c-os>"
  "<c-osversion>" c-osversion "</c-osversion>"
  "<date>" date "</date>"
  "<time>" time "</time>"
  "<c-cpu>" c-cpu "</c-cpu>"
  "<transport>" transport "</transport>"
"</XML>"

c-ip           = "0.0.0.0"
```

The values of the following fields **MUST** be assigned as defined in section [2.1](#) :

- **c-dns**
- **c-os**
- **c-osversion**
- **date**
- **time**
- **c-cpu**
- **transport**

3 Structure Examples

3.1 Legacy Logging Message

The following is an example of a legacy logging message in W3C format:

```
0.0.0.0 2003-09-27 00:27:24 - http://10.194.20.175/mcast1200K 0 42 1
200 {3300AD50-2C39-46c0-AE0A-B4C904C7848E} 9.0.0.2980 en-US
WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008 - wmplayer.exe 9.0.0.2980
Windows_XP 5.1.0.2600 Pentium 1801 268885194 1255347 http TCP
Windows_Media_Audio_9 Windows_Media_Video_9 - - 6321233 - 4496 0 0 0 0
0 0 1 0 100 - - - -
```

The following is an example of a legacy logging message in XML format:

```
<XML>
<Summary>0.0.0.0 2003-09-27 00:27:24 - http://10.194.20.175/mcast1200K 0 42 1 200
{3300AD50-2C39-46c0-AE0A-B4C904C7848E} 9.0.0.2980
en-US WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008 - wmplayer.exe 9.0.0.2980
Windows_XP 5.1.0.2600 Pentium 1801 268885194 1255347
http TCP Windows_Media_Audio_9 Windows_Media_Video_9
- - 6321233 - 4496 0 0 0 0 0 0 1 0 100 - - - -
http://10.194.20.175/mcast1200K?WMBitrate=6000000 30MinTV_1200k_1s_1s_0Q.wmv -
</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2003-09-27</date>
<time>00:27:24</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://10.194.20.175/mcast1200K</cs-uri-stem>
<c-starttime>0</c-starttime>
<x-duration>42</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-B4C904C7848E}</c-playerid>
<c-playerversion>9.0.0.2980</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-Agent>WMFSDK/9.0.0.2980_WMPlayer/9.0.0.3008</cs-User-Agent>
<cs-Referer>-</cs-Referer>
<c-hostexe>wmplayer.exe</c-hostexe>
<c-hostexever>9.0.0.2980</c-hostexever>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>1801</filelength>
<filesize>268885194</filesize>
<avgbandwidth>1255347</avgbandwidth>
<protocol>http</protocol>
<transport>TCP</transport>
<audiocodec>Windows_Media_Audio_9</audiocodec>
<videocodec>Windows_Media_Video_9</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>6321233</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>4496</c-pkts-received>
```

```

<c-pkts-lost-client>0</c-pkts-lost-client>
<c-pkts-lost-net>0</c-pkts-lost-net>
<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>
<c-resendreqs>0</c-resendreqs>
<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>0</c-pkts-recovered-resent>
<c-buffercount>1</c-buffercount>
<c-totalbuffertime>0</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>http://10.194.20.175/mcast1200K?WMBitrate=6000000</cs-url>
<ContentDescription>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL>30MinTV_1200k_1s_1s_0Q.wmv</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_URL>
<WMS_CONTENT_DESCRIPTION_COPIED_METADATA_FROM_PLAYLIST_FILE>1</WMS_CONTENT_DESCRIPTION_COPIED_METADATA_FROM_PLAYLIST_FILE>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_DURATION>1800501</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_DURATION>
<WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_START_OFFSET>1450</WMS_CONTENT_DESCRIPTION_PLAYLIST_ENTRY_START_OFFSET>
<WMS_CONTENT_DESCRIPTION_SERVER_BRANDING_INFO>WMServer/9.0</WMS_CONTENT_DESCRIPTION_SERVER_BRANDING_INFO>
</ContentDescription>
</XML>

```

The following is an example of how a legacy log may appear as sent to a Web server:

```

MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mmsu://foo.microsoft.com/testfile.wma 30 1 1 200 {35301A88-93D3-4F3A-
A284-30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938
Windows_2000 5.0.0.2195 Pentium 225 4551684 1528 mms UDP - - - - 29868
- 4 0 0 0 0 0 0 0 100 172.29.237.102 - - -

```

3.2 Defining Custom Namespaces in an XML Log

An Active Stream Redirector (.asx) file (for more information, see [\[MSDN-WMMETA\]](#)) can be used to append log data to the XML log structure. Vendors may define any number of custom namespaces and name-value pairs in the "client-logging-data" structure, as specified in section [2.4](#), following the Content Description structure.

The following example illustrates how to add the **cs-media-role** field (section [2.1.30](#)) by using an .asx file:

```

<ASX version="3.0">
  <ENTRY>
    <TITLE> My Title </TITLE>
    <Author> My Author </Author>
    <PARAM name="log:cs-media-role" value="Advertisement" />
    <REF href="http://www.foo.MyDomain.com/live" />
  </ENTRY>
</ASX>

```

The additional and/or custom logging information is specified through the use of the PARAM element. To use the PARAM element in this way, the NAME attribute is set to "log:" followed by a log field name and a corresponding VALUE attribute. The log field specified in the NAME attribute is set to the value of the VALUE attribute. If the log does not already contain a field with the specified name, it will be added.

An XML namespace has to be defined for each custom log field specified in an .asx file. This namespace is appended to the NAME attribute and is separated from the field name by a second colon (":"). Because everything after the second colon is treated as a namespace, the field name should not contain a colon.

The following example illustrates the specification of custom log fields using an .asx file:

```
<ASX version="3.0">
  <ENTRY>
    <TITLE> My Title </TITLE>
    <Author> My Author </Author>
    <PARAM name="log:vendor-field1:VendorNameSpace" value="Value1" />
    <PARAM name="log:vendor-field2:VendorNameSpace" value="Value2" />
    <REF href="http://www.foo.MyDomain.com/live" />
  </ENTRY>
</ASX>
```

When an XML log is sent to a server for this .asx file, the new namespace is inserted after the Content Description section, as shown in the following example (many log fields extraneous to this example have been omitted for brevity and clarity):

```
<XML>
  <Summary>0.0.0.0 2003-09-27 00:27:24 ... </Summary>
  <c-ip>0.0.0.0</c-ip>
  <date>2003-09-27</date>
  <time>00:27:24</time>
  ...
  <ContentDescription>
  ...
  </ContentDescription>
  <VendorNameSpace>
    <vendor-field1>Value1</vendor-field1>
    <vendor-field2>Value2</vendor-field2>
  </VendorNameSpace>
</XML>
```

3.3 Example Streaming Log Messages

The following is an example of a [Streaming Log](#) in XML format:

```

<XML>
<Summary>0.0.0.0 2006-05-01 21:34:01 -
http://foo.microsoft.com/content.wmv 4 0 1 200 {3300AD50-2C39-46c0-
AE0A-3E0B6EFB86DC} 10.0.0.3802 en-US
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_(WMFSDK/10.0.0.3802)
_WMPPlayer/10.0.0.4019 http://bar.microsoft.com iexplore.exe
6.0.2900.2180 Windows_XP 5.1.0.2600 Pentium 130 638066 - http TCP - - -
-0 - 0 0 0 0 0 0 0 0 0 100 - - -
http://foo.microsoft.com/content.wmv - -</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2006-05-01</date>
<time>21:34:01</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://foo.microsoft.com/content.wmv</cs-uri-stem>
<c-starttime>4</c-starttime>
<x-duration>0</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-3E0B6EFB86DC}</c-playerid>
<c-playerversion>10.0.0.3802</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-
Agent>Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_(WMFSDK/10.0.0.3802)_WMPPlayer/10
.0.0.4019</cs-User-Agent>
<cs-Referer>http://bar.microsoft.com</cs-Referer>
<c-hostexe>iexplore.exe</c-hostexe>
<c-hostexever>6.0.2900.2180</c-hostexever>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>130</filelength>
<filesize>638066</filesize>
<avgbandwidth>-</avgbandwidth>
<protocol>http</protocol>
<transport>TCP</transport>
<audiocodec>-</audiocodec>
<videocodec>-</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>0</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>0</c-pkts-received>
<c-pkts-lost-client>0</c-pkts-lost-client>
<c-pkts-lost-net>0</c-pkts-lost-net>
<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>
<c-resendreqs>0</c-resendreqs>
<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>0</c-pkts-recovered-resent>
<c-buffercount>0</c-buffercount>
<c-totalbuffertime>0</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>http://foo.microsoft.com/content.wmv</cs-url>
<cs-media-name>-</cs-media-name>
<cs-media-role>-</cs-media-role>

```

</XML>

The following is an example of how a Streaming Log may appear as sent to a Web server:

```
MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mmsu://foo.microsoft.com/testfile.wma 30 1 1 200 {35301A88-93D3-4F3A-
A284-30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938
Windows_2000 5.0.0.2195 Pentium 225 4551684 1528 mms UDP - - - - 29868
- 4 0 0 0 0 0 0 0 100 172.29.237.102 - - -
mmsu://foo.microsoft.com/testfile.wma - -
```

3.4 Example Rendering Log Messages

The following is an example of a [Rendering Log](#) in XML format:

```
<XML>
<Summary>0.0.0.0 2006-05-01 21:34:01 -
http://foo.microsoft.com/content.wmv 4 0 1 200 {3300AD50-2C39-46c0-
AE0A-3E0B6EFB86DC} 10.0.0.3802 en-US
Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)_(WMFSDK/10.0.0.3802)
_WMPlayer/10.0.0.4019 http://bar.microsoft.com iexplore.exe
6.0.2900.2180 Windows_XP 5.1.0.2600 Pentium 130 638066 - Cache -
Windows_Media_Audio_9 Windows_Media_Video_9 - - 0 - - - - - - - -
100 - - - - - - http://foo.microsoft.com/content.wmv - - - 0
</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2006-05-01</date>
<time>21:34:01</time>
<c-dns>-</c-dns>
<cs-uri-stem>http://foo.microsoft.com/content.wmv</cs-uri-stem>
<c-starttime>4</c-starttime>
<x-duration>0</x-duration>
<c-rate>1</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-3E0B6EFB86DC}</c-playerid>
<c-playerversion>10.0.0.3802</c-playerversion>
<c-playerlanguage>en-US</c-playerlanguage>
<cs-User-Agent>Mozilla/4.0_(compatible;_MSIE_6.0;_Windows_NT_5.1)
_(WMFSDK/10.0.0.3802)_WMPlayer/10.0.0.4019</cs-User-Agent>
<cs-Referer>http://bar.microsoft.com</cs-Referer>
<c-hostexe>iexplore.exe</c-hostexe>
<c-hostexeversion>6.0.2900.2180</c-hostexeversion>
<c-os>Windows_XP</c-os>
<c-osversion>5.1.0.2600</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>130</filelength>
<filesize>638066</filesize>
<avgbandwidth>-</avgbandwidth>
<protocol>Cache</protocol>
<transport>-</transport>
<audiocodec>Windows_Media_Audio_9</audiocodec>
<videocodec>Windows_Media_Video_9</videocodec>
```

```

<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>0</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>-</c-pkts-received>
<c-pkts-lost-client>-</c-pkts-lost-client>
<c-pkts-lost-net>-</c-pkts-lost-net>
<c-pkts-lost-cont-net>-</c-pkts-lost-cont-net>
<c-resendreqs>-</c-resendreqs>
<c-pkts-recovered-ECC>-</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>-</c-pkts-recovered-resent>
<c-buffercount>-</c-buffercount>
<c-totalbuffertime>-</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>http://foo.microsoft.com/content.wmv</cs-url>
<cs-media-name>-</cs-media-name>
<cs-media-role>-</cs-media-role>
</XML>

```

The following is an example of how a Rendering Log may appear as sent to a Web server:

```

MX_STATS_LogLine: 0.0.0.0 2000-06-14 01:18:58 -
mms://foo.microsoft.com/test.wma 30 1 1 200 {35301A88-93D3-4F3A-A284-
30F7A611CD23} 7.0.0.1938 en-US - - wmplayer.exe 7.0.0.1938 Windows_2000
5.0.0.2195 Pentium 225 4551684 1528 Cache - - - - 29868 - - - - -
- - - 100 - - - - mms://foo.microsoft.com/test.wma - -

```

3.5 Example Connect-Time Log Message

The following is an example of a [Connect-Time Log](#) message in XML format:

```

<XML>
<Summary>- 0.0.0.0 Windows 6.0.0.6000 2006-08-30 13:18:44 Pentium
TCP</Summary>
<c-dns>-</c-dns>
<c-ip>0.0.0.0</c-ip>
<c-os>Windows</c-os>
<c-osversion>6.0.0.6000</c-osversion>
<date>2006-08-30</date>
<time>13:18:44</time>
<c-cpu>Pentium</c-cpu>
<transport>TCP</transport>
</XML>

```

3.6 Example Log Sent to a Web Server

The following is an example of a client validating a logging URL and subsequently transmitting a logging message to the Web server:

```
GET /scripts/wmsiislog.dll HTTP/1.1
User-Agent: NSPlayer
Host: WebServer:8080
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Connection: close
Date: Wed, 27 Jun 2007 02:54:23 GMT
Server: Microsoft-IIS/6.0
Content-Type: text/html

<head><title>WMS ISAPI Log Dll/9.00.00.3372</title></head>
<body><h1>WMS ISAPI Log Dll/9.00.00.3372</h1></body>

POST /scripts/wmsiislog.dll HTTP/1.1
Content-Type: text/plain;charset=UTF-8
User-Agent: NSPlayer
Host: WebServer:8080
Content-Length: 424
Connection: Keep-Alive
Cache-Control: no-cache

MX_STATS_LogLine: .0.0.0.0 2007-06-27 02:52:39 - asfm://239.192.50.29:30864 0 39 1 200
{3300AD50-2C39-46c0-AE0A-0572F2EA5330} 10.0.0.4054 en-US
WMFSDK/10.0.0.4054_WMPPlayer/10.0.0.4036 - wmpplayer.exe 10.0.0.3802 Windows_XP 5.1.0.2600
Pentium 229 10413011 411536 asfm UDP Windows_Media_Audio_9.2 -
http://WebServer:8080/multicast.nsc - 2170350 - 182 0 0 0 0 0 1 3 100 239.192.50.29 - -
- http://WebServer:8080/multicast.nsc - -
HTTP/1.1 200 OK
Server: Microsoft-IIS/6.0
Date: Wed, 27 Jun 2007 02:54:23 GMT
Connection: close
```

3.7 Parsing Windows Media Log Files

Microsoft Log Parser 2.2 is a tool that queries text-based data and other system data sources, including Windows Media log files. For more information, see [\[MSFT-LOGPARSER\]](#).

4 Security Considerations

A server that receives a logging message SHOULD validate the syntax of the fields. For example, the server should check that logging fields that are supposed to contain numerical data really do so, and that no invalid characters, such as control characters, are present. Invalid fields or characters could cause any tools that process the logging information to malfunction.

5 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

[<1> Section 2.1.5:](#) Windows Media Player 6.4 specifies the DNS name in the **c-dns** field.

[<2> Section 2.1.10:](#) On Windows Vista, **c-os** is set to "Windows".

[<3> Section 2.1.25.2:](#) Windows Media Player 6.4, Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP never specify status code 210.

[<4> Section 2.5.1:](#) Windows Media Player 6.4 specifies its own IP address in the **c-ip** field. Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP specify their own IP address in the **c-ip** field depending on the current setting of a configuration value in the user interface.

[<5> Section 2.5.1:](#) Windows Media Player 6.4, Windows Media Format 7.0 SDK, Windows Media Format 7.1 SDK, and Windows Media Player for Windows XP never include the three optional fields.

[<6> Section 2.5.3:](#) Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista do not include the "contentdescription" and "client-logging-data" syntax elements in the XML-format logging message when using RTSP [\[MS-RTSP\]](#).

[<7> Section 2.6.2:](#) Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista do not include the "contentdescription" and "client-logging-data" syntax elements in the XML-format logging message when using RTSP [\[MS-RTSP\]](#).

6 Index

A

[Applicability](#)
[audiocodec](#)
[avgbandwidth](#)

B

[Basic logging syntax](#)

C

[c-buffercount](#)
[c-bytes](#)
[c-channelURL](#)
[c-cpu](#)
[c-dns](#)
[c-hostexe](#)
[c-hostexever](#)
[c-ip \(section 2.1.8, section 2.1.9\)](#)
[Connect-time log](#)
[Connect-time logging syntax](#)
[c-os](#)
[c-osversion](#)
[c-pkts-lost-client](#)
[c-pkts-lost-cont-net](#)
[c-pkts-lost-net](#)
[c-pkts-received](#)
[c-pkts-recovered-ECC](#)
[c-pkts-recovered-resent](#)
[c-playerid](#)
[c-playerlanguage](#)
[c-playerversion](#)
[c-quality](#)
[c-rate](#)
[c-resendreqs](#)
[cs-media-name](#)
[cs-media-role](#)
[cs-Referer](#)
[c-starttime](#)
[c-status](#)
[cs-uri-stem](#)
[cs-uri](#)
[cs-User-Agent](#)
[cs-user-name](#)
[c-totalbuffertime](#)

D

[date](#)

E

Examples
[legacy logging message example](#)
[overview](#)
[parsing Windows Media log files example](#)
[Extended logging syntax](#)

F

[Fields - vendor-extensible](#)
[filelength](#)
[filesize](#)

G

[Glossary](#)

I

[Informative references](#)
[Introduction](#)

L

Legacy log
[common definitions](#)
[overview](#)
[sent to Web server](#)
[W3C format](#)
[XML format](#)
[Legacy logging message example](#)
[Localization](#)
[Log data fields](#)
[Logging message - W3C syntax](#)
[Logging message - WXML schema](#)
[Logging message sent to Web servers](#)

N

[Normative references](#)

P

[Parsing Windows Media log files example](#)
[protocol](#)

R

References
[informative](#)
[normative](#)
[overview](#)
[Relationship to other protocols](#)
Rendering log
[common definitions](#)
[overview](#)
[sent to Web server](#)
[sent to Windows Media Services](#)

S

[sc-bytes](#)
[s-content-path](#)
[s-cpu-util](#)

[s-dns](#)
[Security](#)
[s-ip](#)
[s-pkts-sent](#)
[s-proxied](#)
[s-session-id](#)
[Status Code 200 \(No Error\)](#)
[Status Code 210 \(Client Successfully Reconnected\)](#)
[s-totalclients](#)
Streaming log
 [common definitions](#)
 [overview](#)
 [sent to Web server](#)
 [sent to Windows Media Services](#)
Structures
 [connect-time log](#)
 [legacy log](#)
 [log data fields](#)
 [logging message - W3C syntax](#)
 [logging message - XML schema](#)
 [logging message sent to Web servers](#)
 [overview](#)
 [rendering log](#)
 [streaming log](#)

T

[time](#)
[transport](#)

V

[Vendor-extensible fields](#)
[Versioning](#)
[videocodec](#)

W

[Windows behavior](#)

X

[x-duration](#)

APPENDIX B-7



US007089259B1

(12) **United States Patent**
Kouznetsov et al.

(10) **Patent No.:** **US 7,089,259 B1**
(45) **Date of Patent:** **Aug. 8, 2006**

(54) **SYSTEM AND METHOD FOR PROVIDING A FRAMEWORK FOR NETWORK APPLIANCE MANAGEMENT IN A DISTRIBUTED COMPUTING ENVIRONMENT**

(75) Inventors: **Victor Kouznetsov**, Aloha, OR (US);
Michael Chin-Hwan Pak, Portland, OR (US); **Daniel J. Melchione**, Beaverton, OR (US); **Ian Shaughnessy**, Portland, OR (US)

| | | | | |
|--------------|------|---------|----------------|---------|
| 5,978,912 | A * | 11/1999 | Rakavy et al. | 713/2 |
| 6,123,737 | A * | 9/2000 | Sadowsky | 717/173 |
| 6,256,668 | B1 * | 7/2001 | Slivka et al. | 709/220 |
| 6,345,294 | B1 * | 2/2002 | O'Toole et al. | 709/222 |
| 6,658,585 | B1 * | 12/2003 | Levi | 714/4 |
| 6,757,723 | B1 * | 6/2004 | O'Toole et al. | 709/222 |
| 6,993,657 | B1 * | 1/2006 | Renner et al. | 713/182 |
| 2002/0184619 | A1 * | 12/2002 | Meyerson | 717/173 |
| 2003/0028624 | A1 * | 2/2003 | Hasan et al. | 709/220 |
| 2006/0031454 | A1 * | 2/2006 | Ewing et al. | 709/223 |

OTHER PUBLICATIONS

(73) Assignee: **McAfee, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 362 days.

“Understanding UPnP (TM): A Whitepaper” Jun. 2000, UPnP (TM) Forum, 39 pages, available online at <http://www.upnp.org/resources/whitepapers.asp>.*
“Plug-In Guide” Jan. 1998, Netscape Communications (TM), TOC has 4 pages, Chapter 1 has 14 pages, Reference has 19 pages, available online at <http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm>.*

(21) Appl. No.: **10/056,702**

(22) Filed: **Jan. 25, 2002**
(Under 37 CFR 1.47)

* cited by examiner

Related U.S. Application Data

(60) Provisional application No. 60/309,835, filed on Aug. 3, 2001, provisional application No. 60/309,858, filed on Aug. 3, 2001.

Primary Examiner—Thuy N. Pardo
(74) *Attorney, Agent, or Firm*—Zilka-Kotab, PC; Christopher J. Hamaty

(51) **Int. Cl.**
G06F 17/30 (2006.01)
(52) **U.S. Cl.** **707/102**; 707/101; 707/103 X; 707/104.1; 709/220; 709/222; 709/223
(58) **Field of Classification Search** 709/220, 709/222, 223; 717/173; 707/101, 102, 103 X, 707/104.1; 714/4; 713/182
See application file for complete search history.

(57) **ABSTRACT**

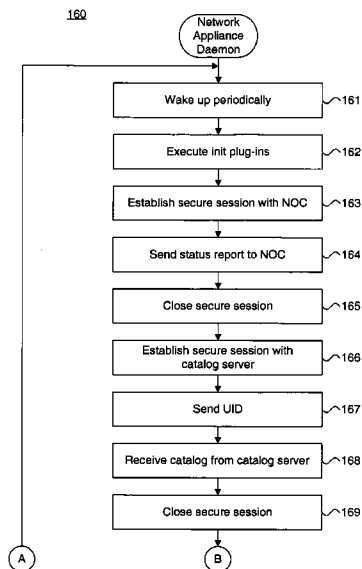
A system and method for providing a framework for network appliance management in a distributed computing environment is disclosed. A status report periodically received from each of a plurality of network appliances is recorded. Each status report contains health and status information and application-specific data for each network appliance. Configuration settings for each network appliance progressively assembled concurrent to providing installable components are maintained. A catalog listing currently installable components for each network appliance based on the configuration settings is dynamically provided.

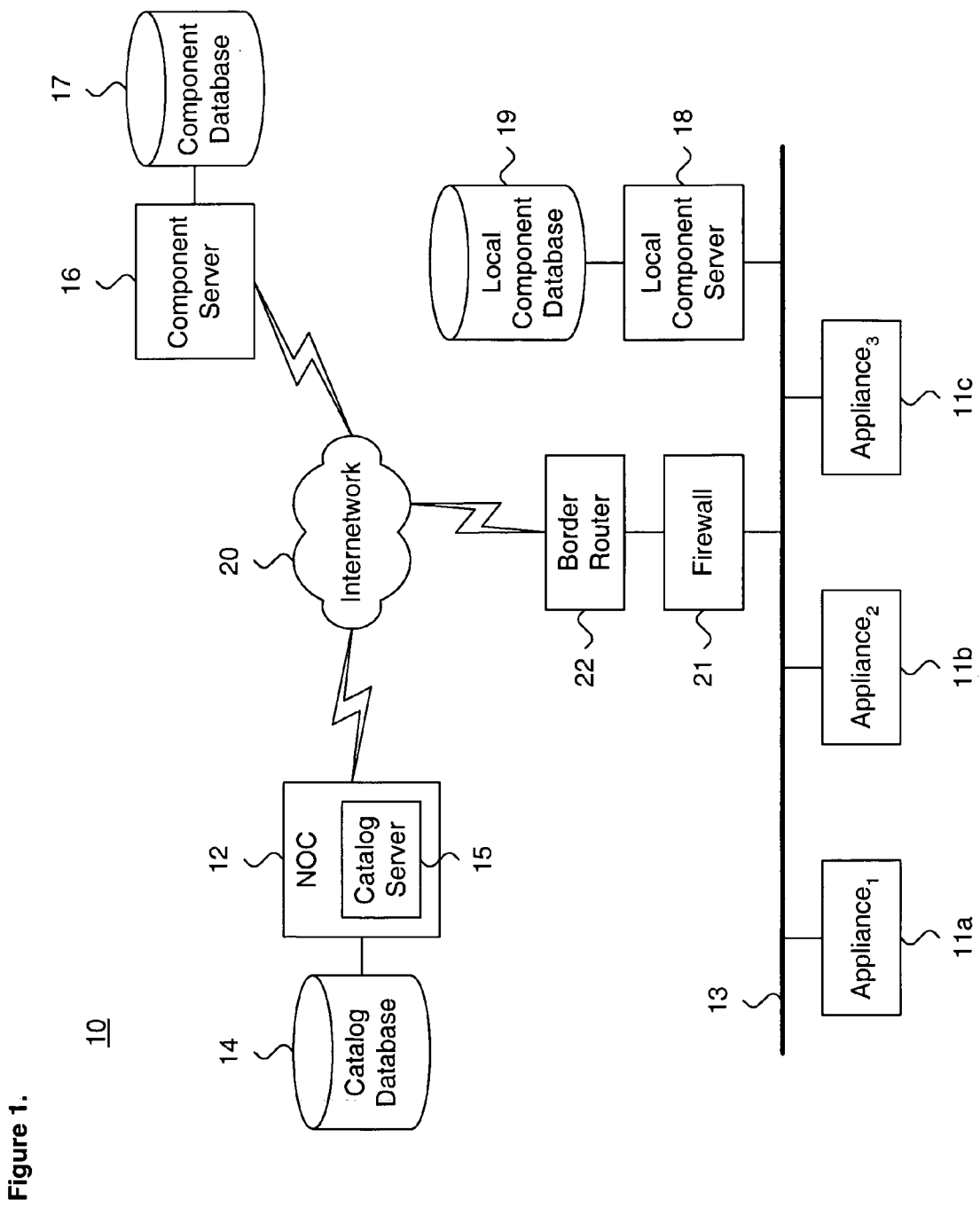
(56) **References Cited**

U.S. PATENT DOCUMENTS

5,974,454 A * 10/1999 Apfel et al. 709/221

47 Claims, 14 Drawing Sheets





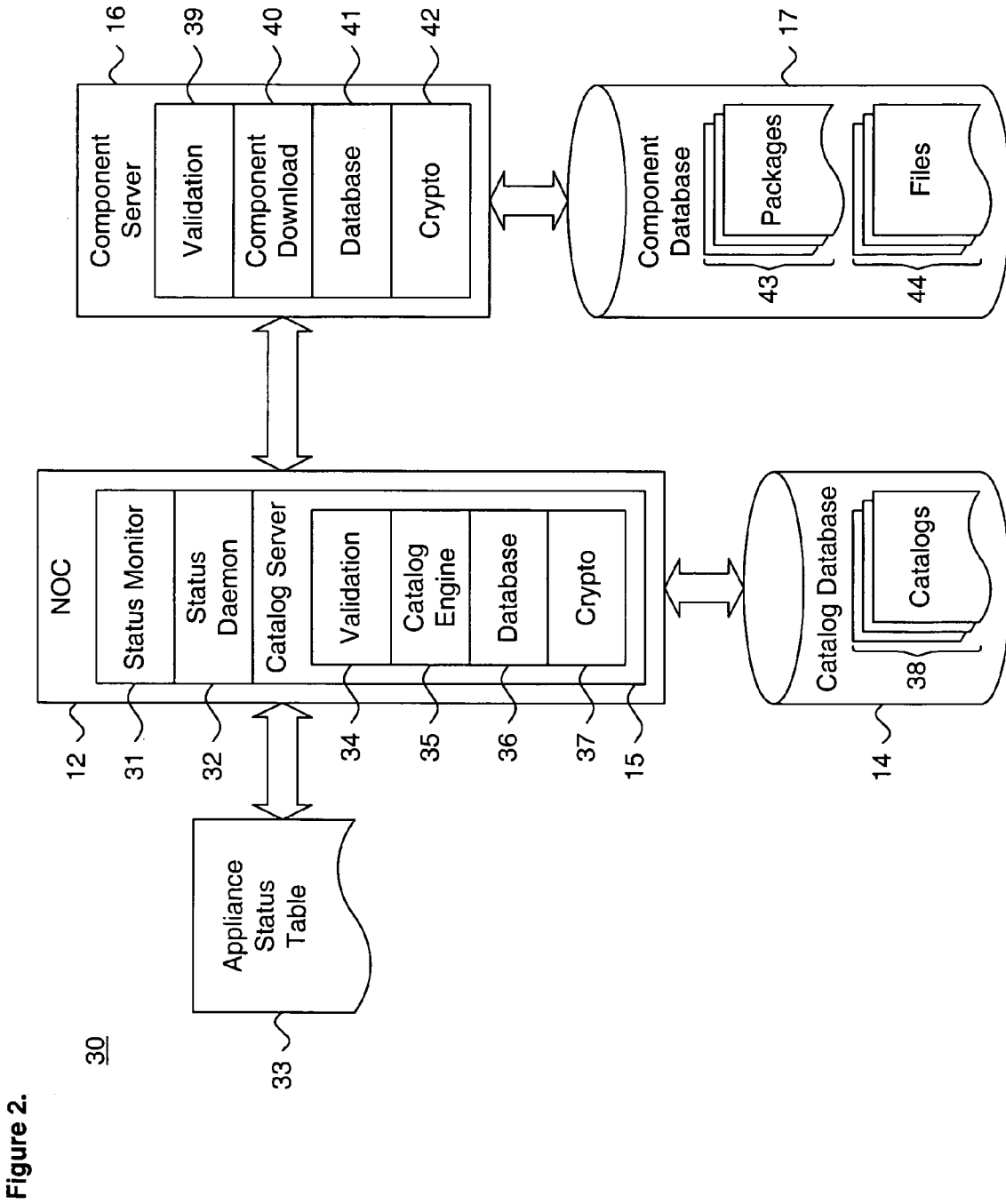
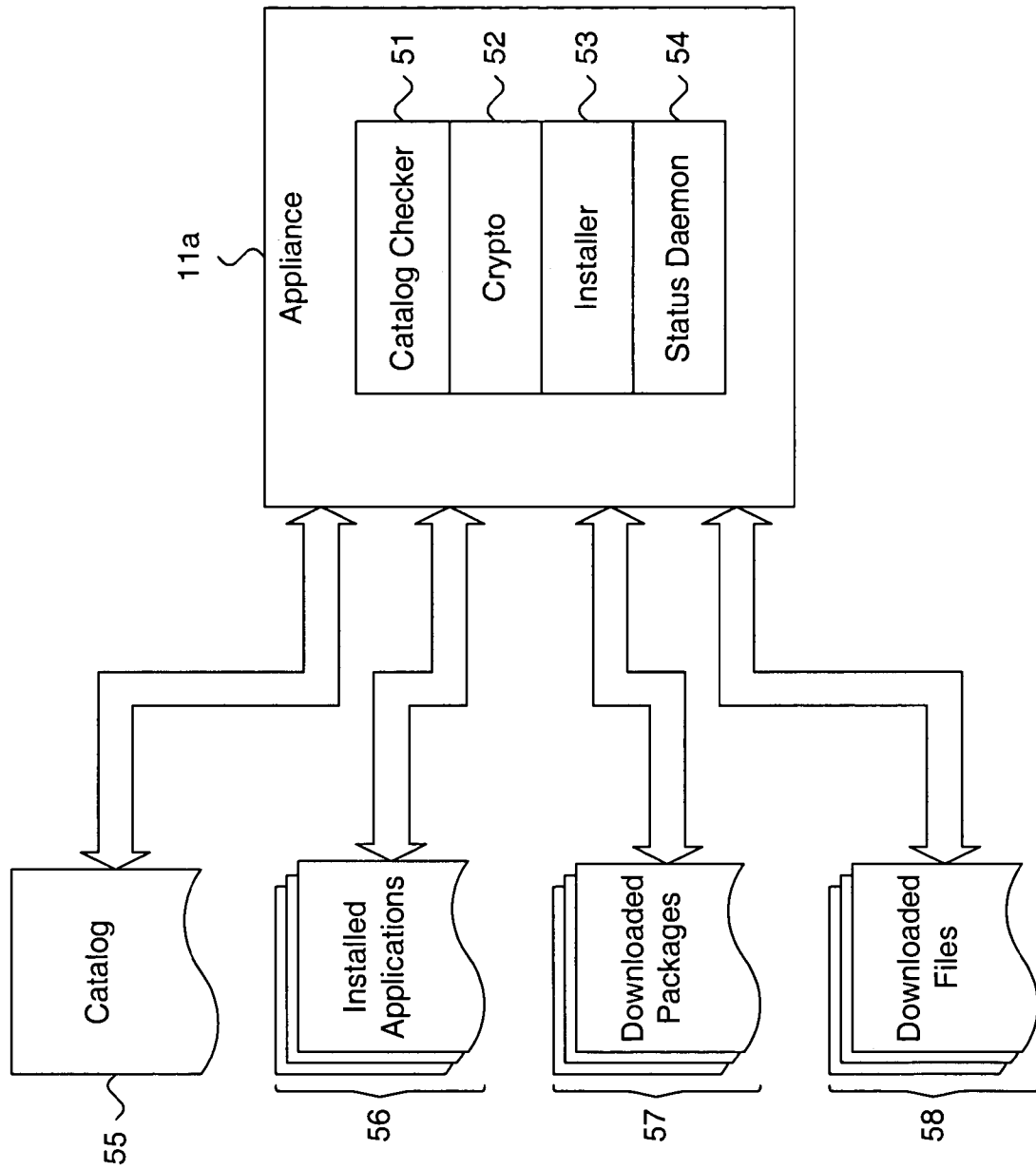


Figure 2.



50

Figure 3.

Figure 4.

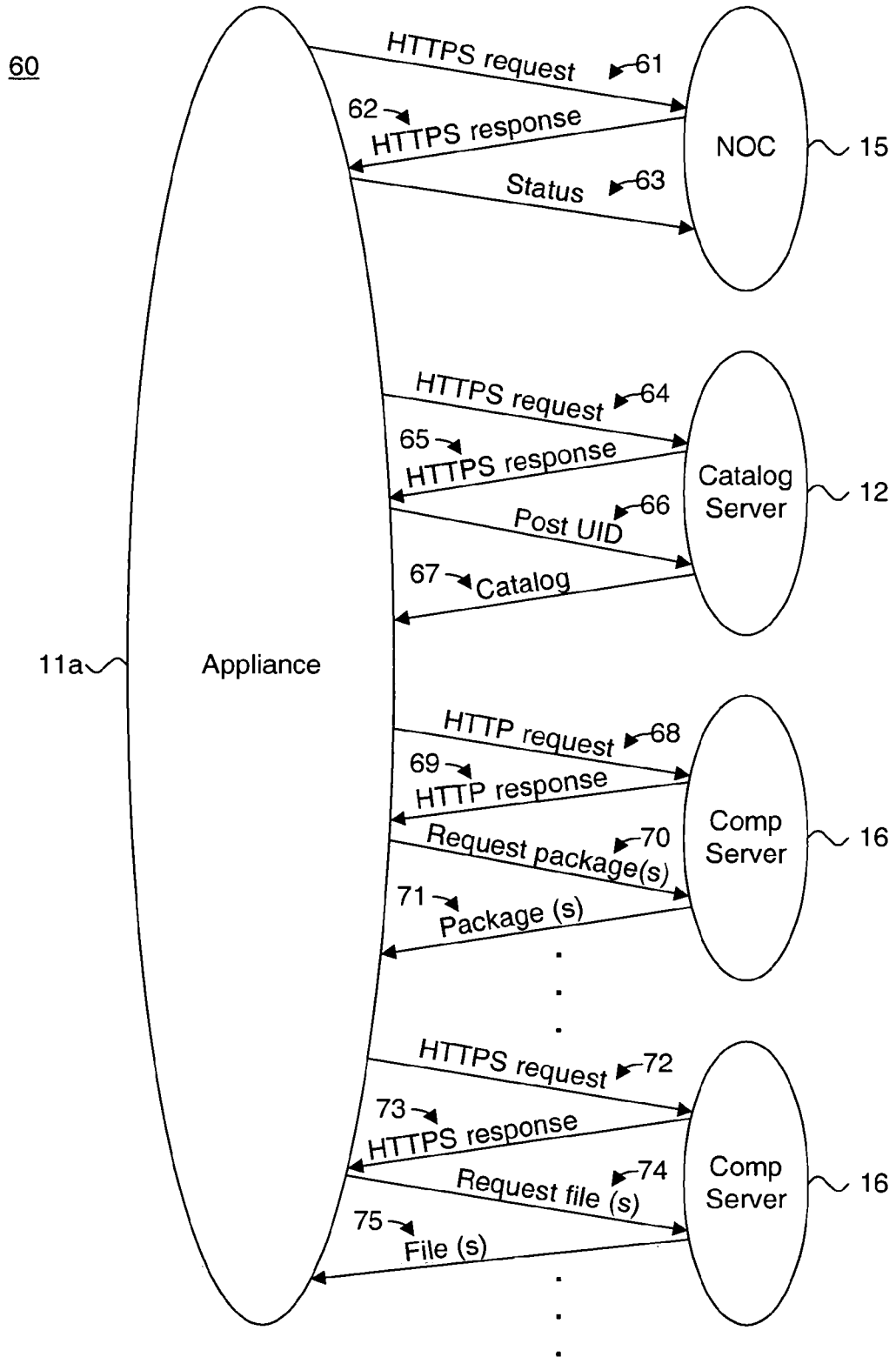


Figure 5.

80

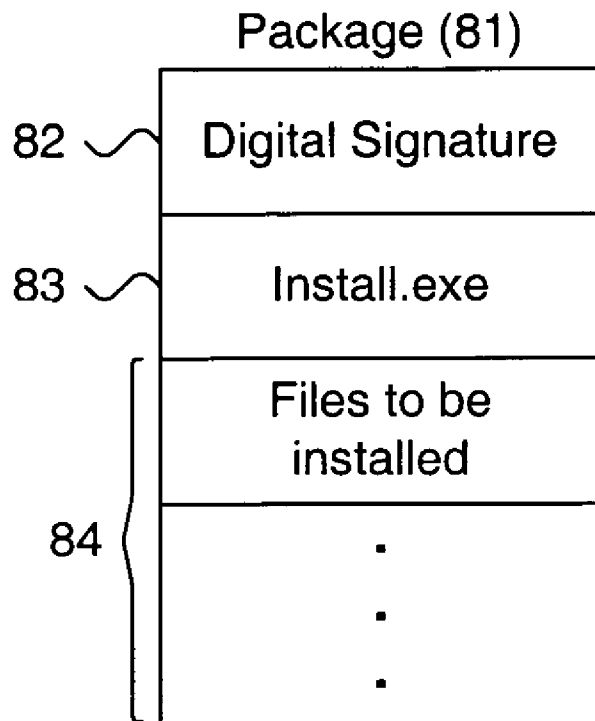


Figure 6.

100

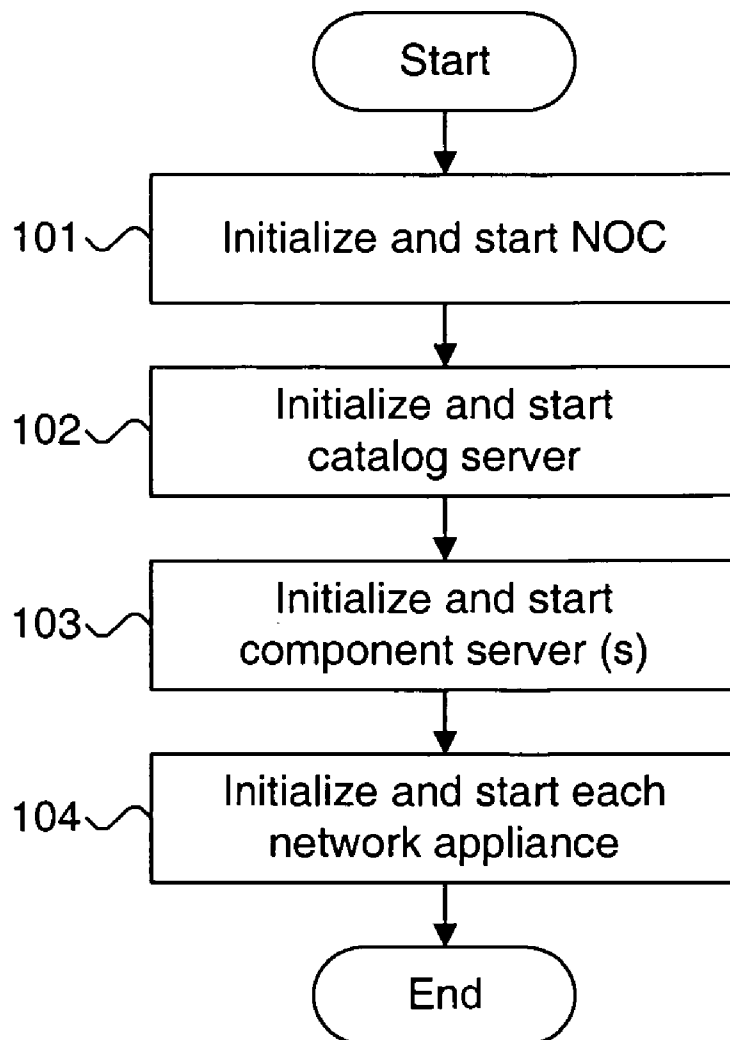


Figure 7.

110

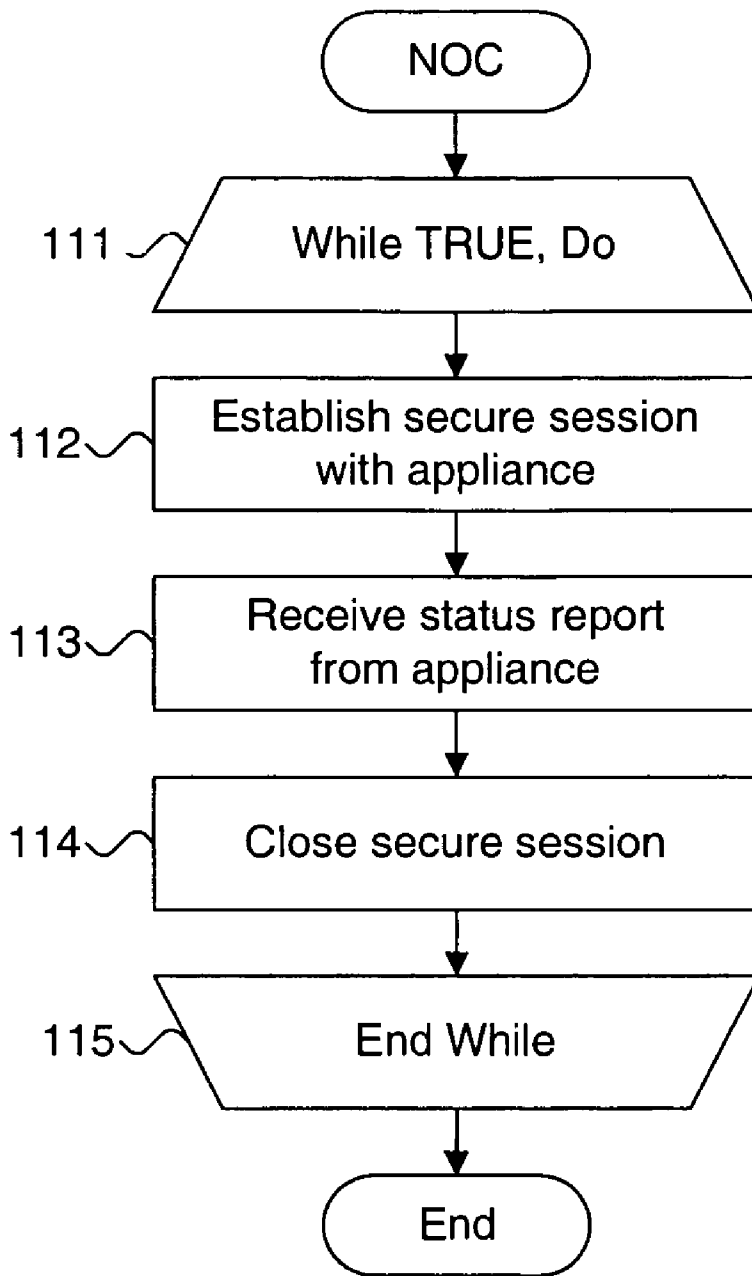


Figure 8.

120

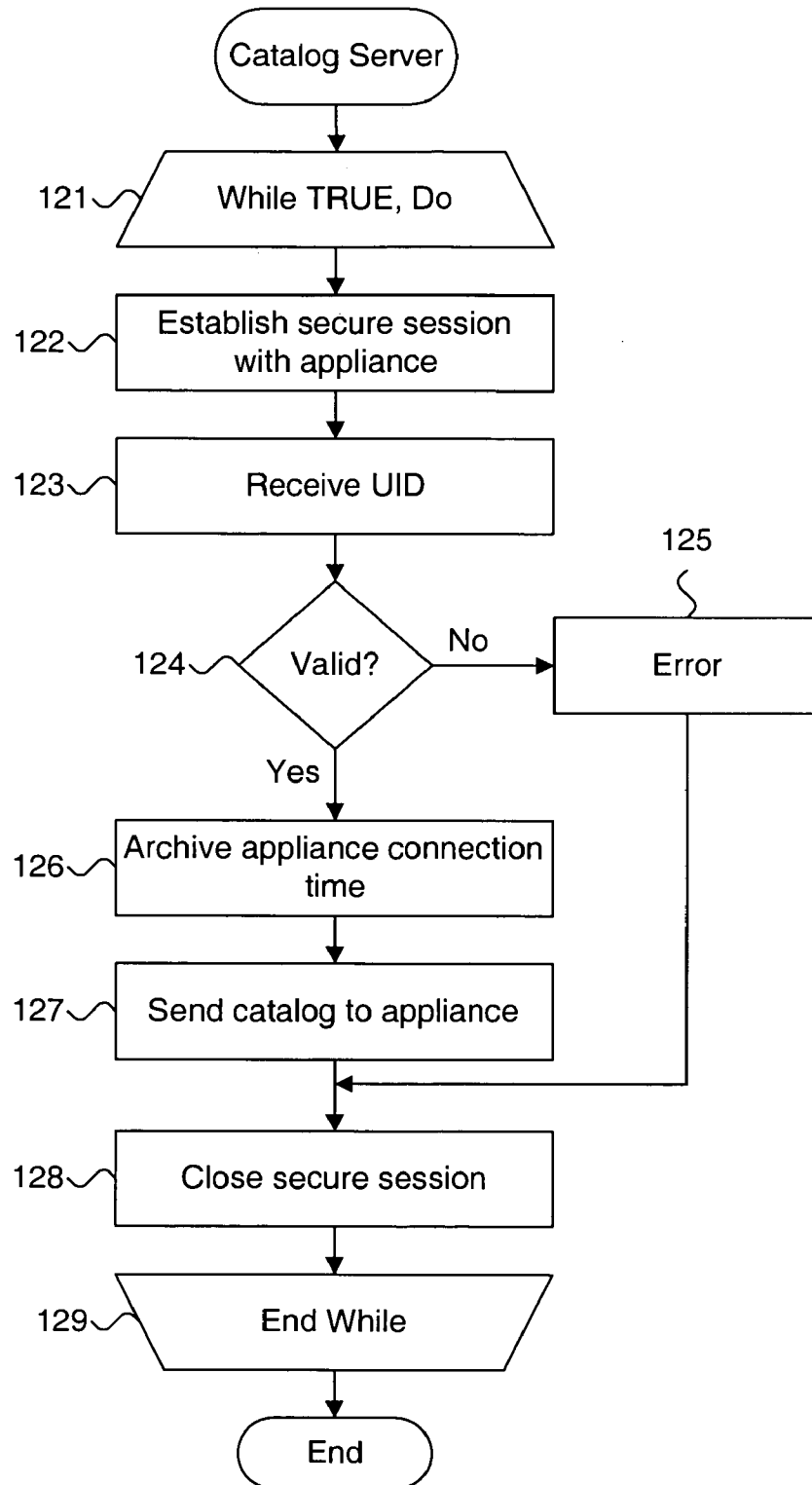


Figure 9.

140

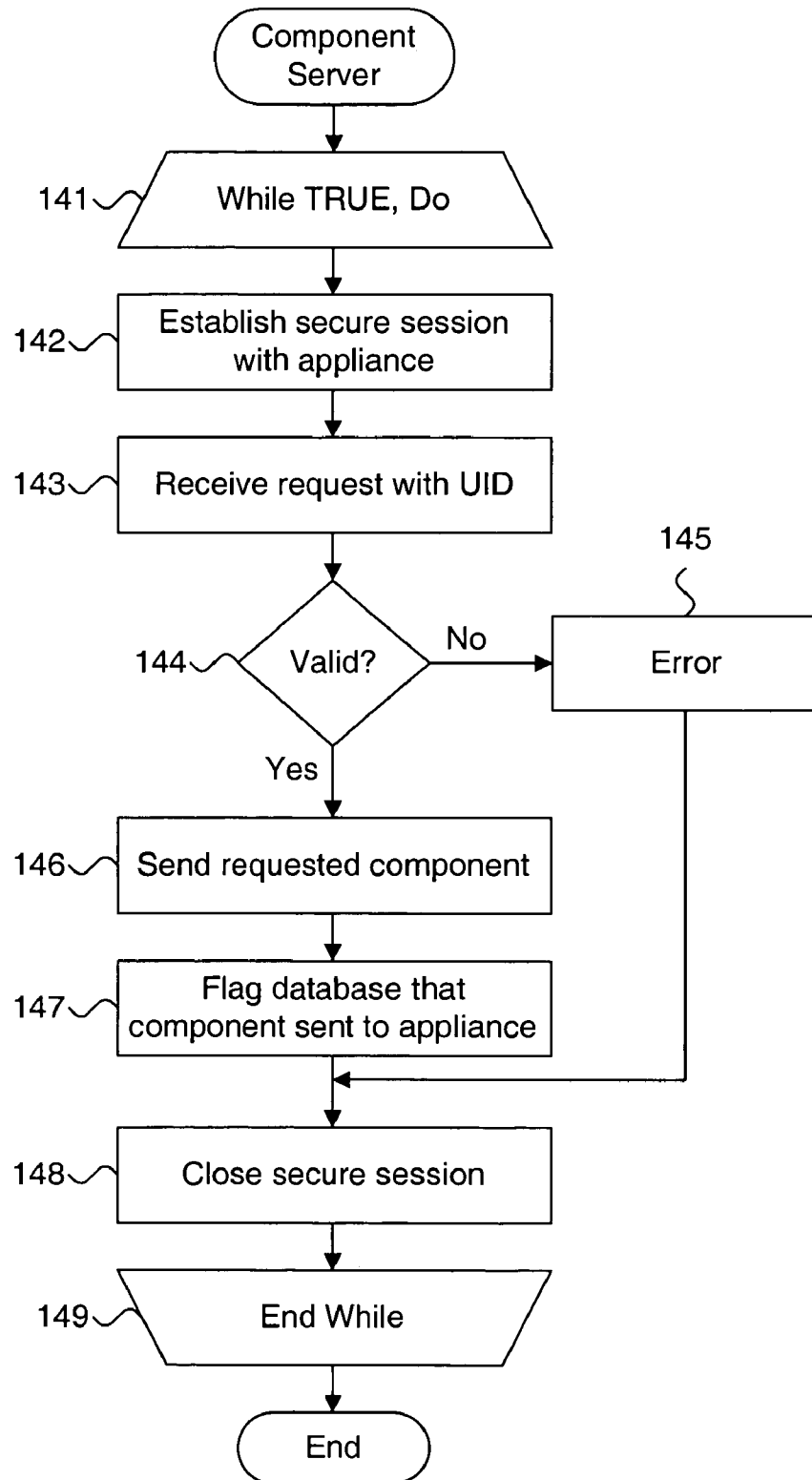


Figure 10A.

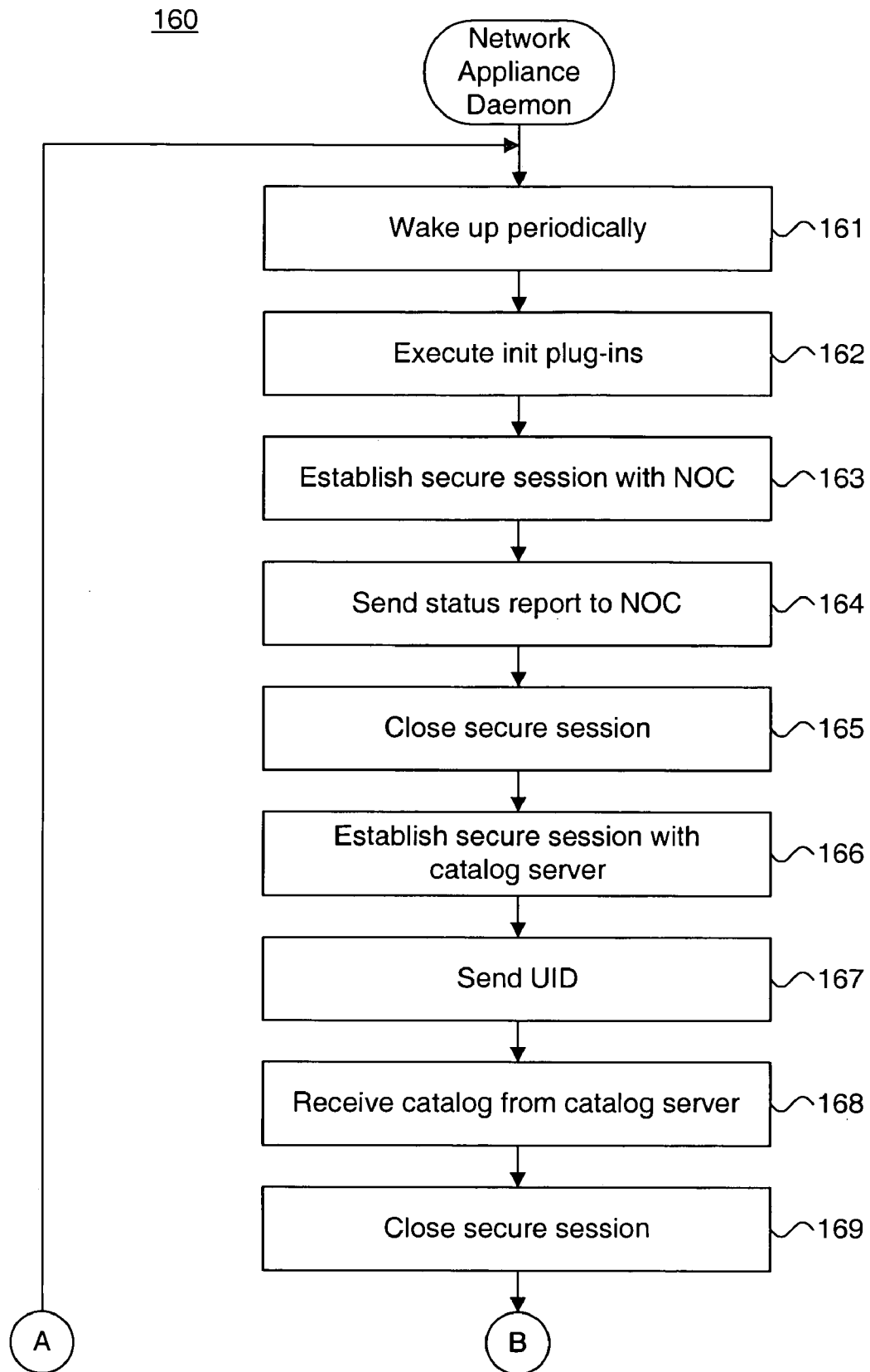


Figure 10B.

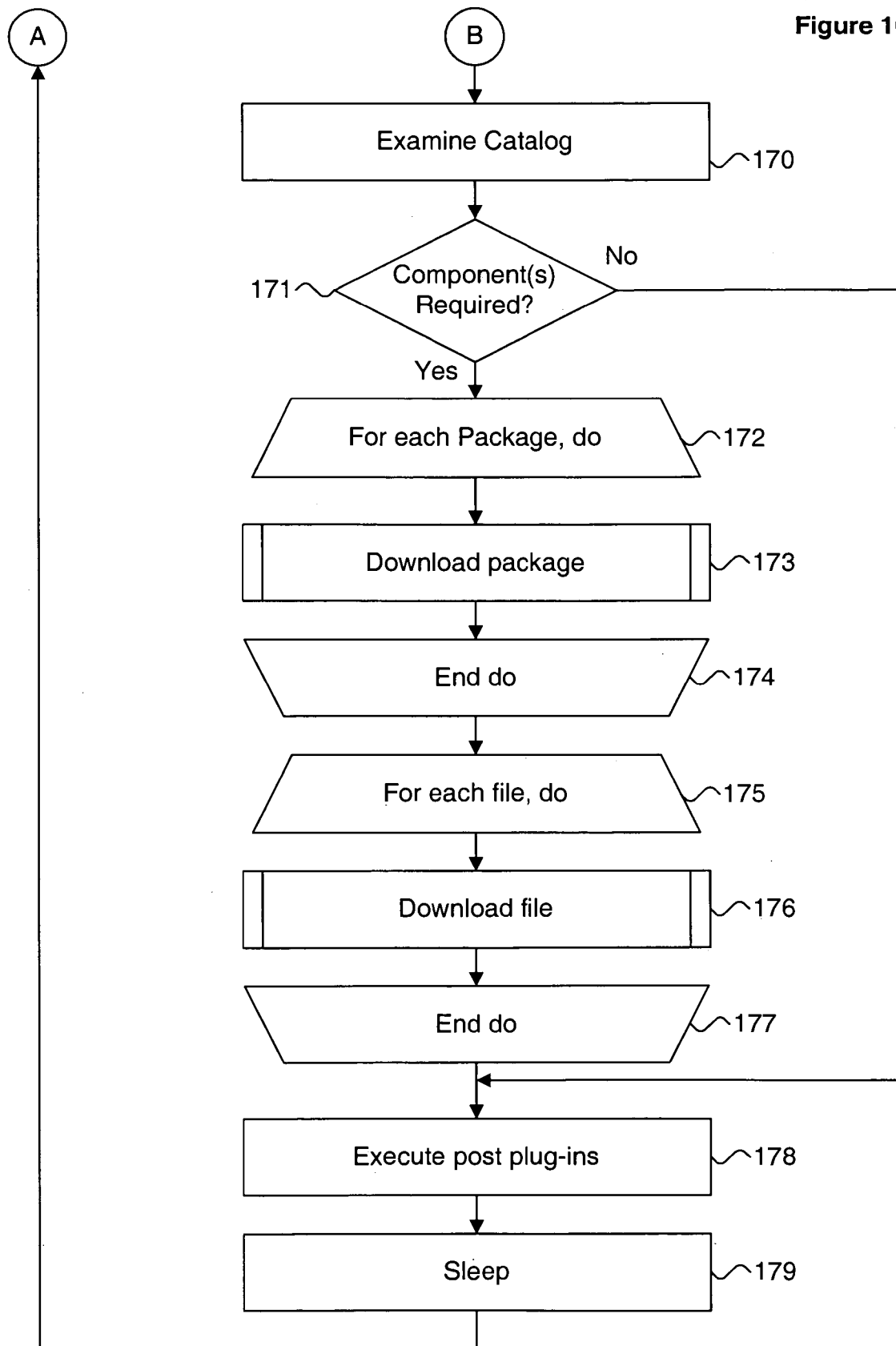


Figure 11.

180

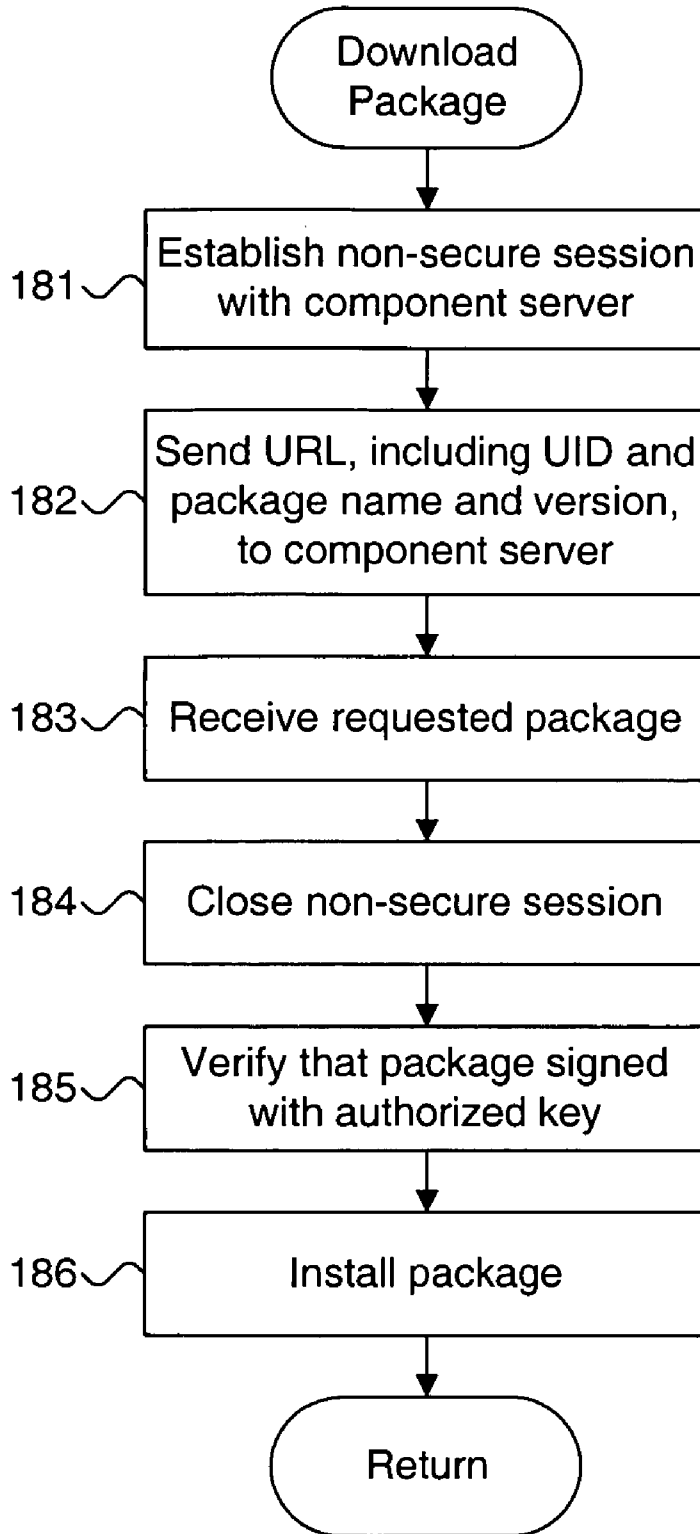


Figure 12.

190

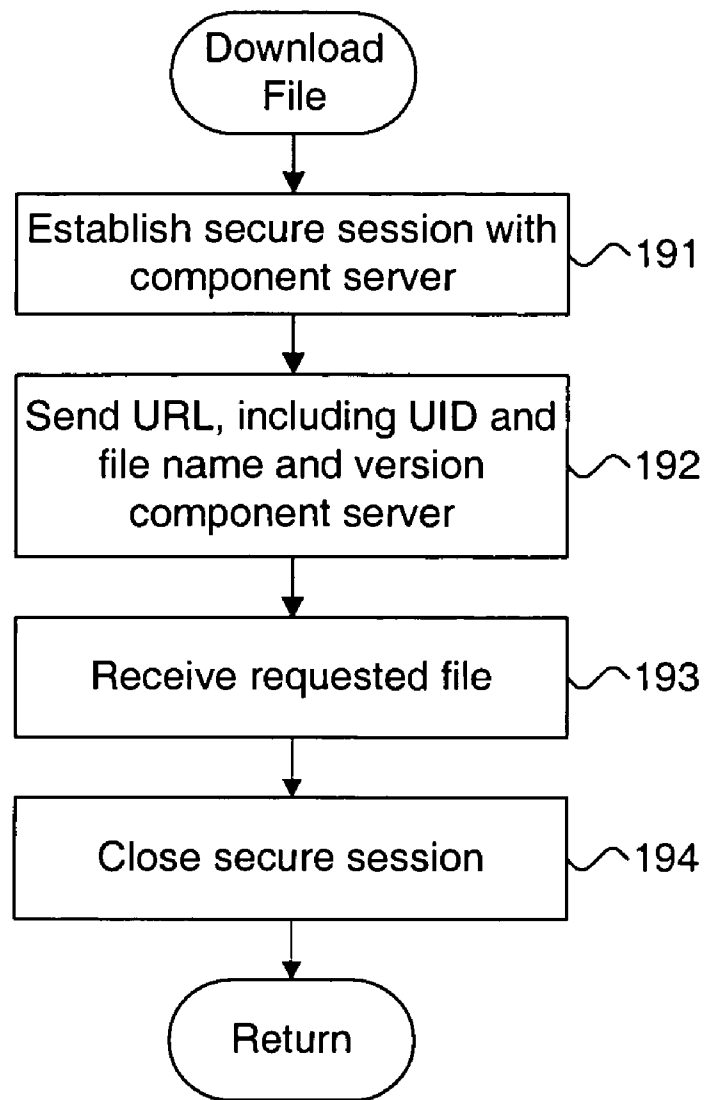
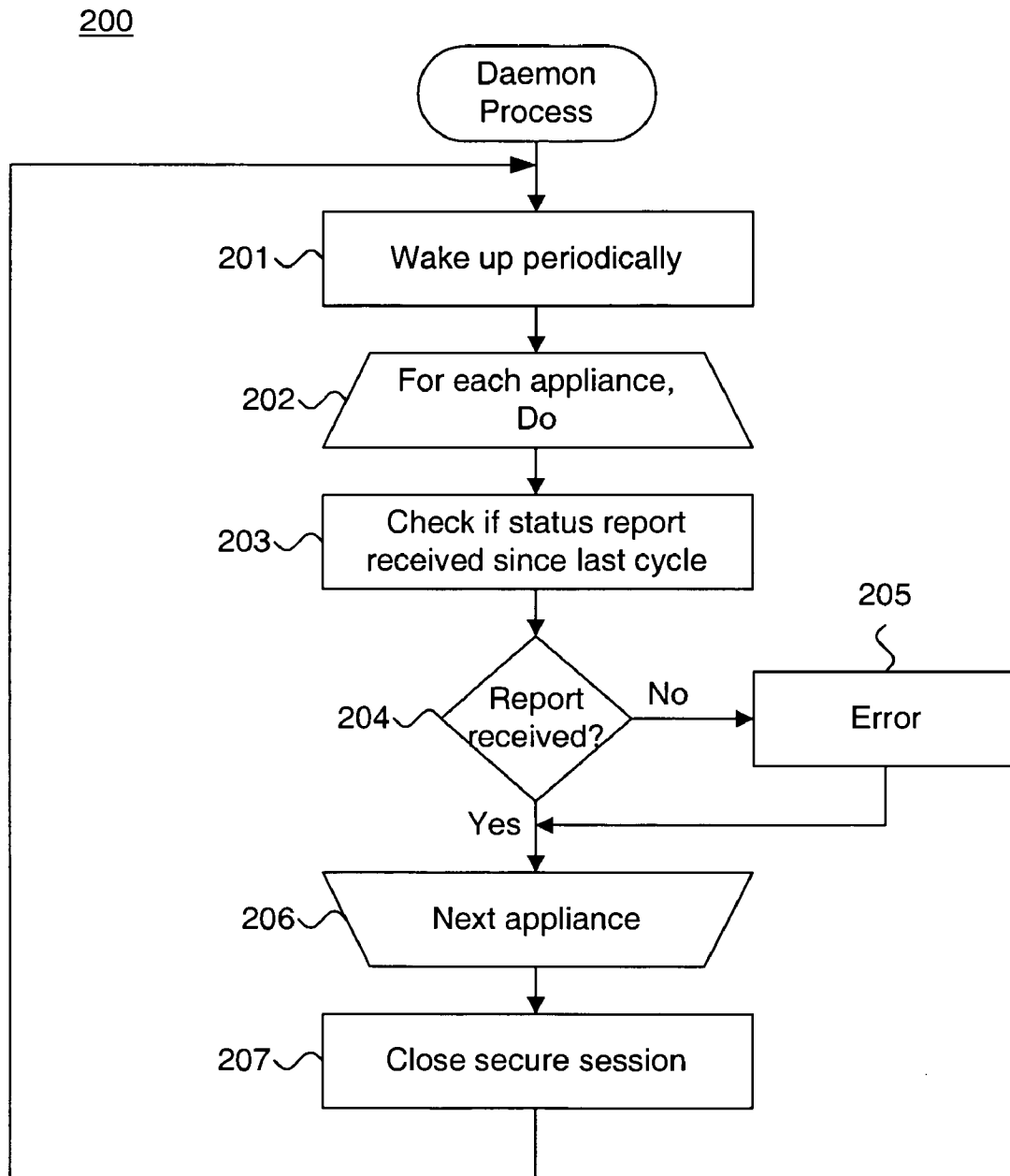


Figure 13.



**SYSTEM AND METHOD FOR PROVIDING A
FRAMEWORK FOR NETWORK APPLIANCE
MANAGEMENT IN A DISTRIBUTED
COMPUTING ENVIRONMENT**

CROSS-REFERENCE TO RELATED
APPLICATIONS

This patent application is a conversion of U.S. provisional patent applications, Ser. No. 60/309,835, filed Aug. 3, 2001, pending; and Ser. No. 60/309,858, filed Aug. 3, 2001, pending; the priority dates of which are claimed and the disclosures of which are incorporated by reference.

FIELD OF THE INVENTION

The present invention relates in general to secure network appliance management and, in particular, to a system and method for providing a framework for network appliance management in a distributed computing environment.

BACKGROUND OF THE INVENTION

Enterprise computing environments generally include both localized intranetworks of interconnected computer systems and resources internal to an organization and geographically distributed internetworks, including the Internet. Intranetworks make legacy databases and information resources available for controlled access and data exchange. Internetworks enable internal users to access remote data repositories and computational resources and allow outside users to access select internal resources for completing limited transactions or data transfer.

Increasingly, network appliances, or simply "appliances," are being deployed within intranetworks to compliment and extend the types of services offered. As a class, network appliances have closed architectures and often lack a standard user interface. These devices provide specialized services, such as electronic mail (email) anti-virus scanning, content filtering, file, Web and print service, and packet routing functions.

Ideally, network appliances should be minimal maintenance devices, which are purchased, plugged into a network, and put into use with no further modification or change. Analogous to a cellular telephone, a network appliance should ideally provide the service promised without requiring active management by individual users or administrators.

Nevertheless, regular maintenance of networks appliance is necessary to ensure continued optimal performance. Operating system and application programs must be installed upon appliance installation and following any type of crash or abnormal service termination. As well, each appliance must be configured, preferably automatically, to comply with applicable security and administration policies. Moreover, as bug fixes and enhancements become available, installed programs must be updated with patches, which must first be obtained from the appropriate sources and then installed on each individual device.

One common problem in maintaining network appliances is the increased workload imposed on individual servers to support appliance maintenance. The health and status of each appliance must be regularly monitored by a server to ensure proper performance and function. Accordingly, individual server loads increase with the addition of each new appliance. The tracking and management of configurations of individual appliances can become resource intensive,

particularly in a large scale network environment containing numerous network appliances.

In the prior art, "push" solutions have been used to manage individual network appliances, whereby changes in configurations and programs are sent to individual appliances from a centralized server as necessary. The server stores each appliance configuration and lists names and versions of programs installed. Periodically, the server polls the pool of appliances to ascertain status and health and pushes new updates out to individual appliances as necessary. However, push solutions are resource intensive and can exact a high performance load on each server. Moreover, servers can fail to detect misconfigurations of appliances erroneously tracked with incorrect configurations.

Therefore, there is a need for an approach to providing autonomous network appliance configuration and management without requiring an active centralized server. Preferably, such an approach would utilize "pull" downloads of needed updates and would further lodge configuration and management responsibilities on individual appliances.

There is a further need for an approach to maintaining the health and status of individual appliances through periodic client-centric reporting. Preferably, such an approach would use a secure "heartbeat" automatically generated by individual appliances to report configuration and status information. As well, each responsible server would preferably generate an alert whenever a heartbeat report was not timely received.

There is a further need for an approach to providing distributed staging of program updates for network appliances. Preferably, such an approach would provide centralized component download management with the capability to instruct requesting appliances to redirect and download software updates from proxy component servers.

SUMMARY OF THE INVENTION

The present invention provides a system and method for autonomously managing the configuration of network appliances deployed in a distributed network environment. Each network appliance executes an installed set of packages and files. Periodically, the appliance awakens to send a report to a network operations center to describe the current health and status of the appliance and provide application-specific data. The network appliance then obtains a catalog of up-to-date packages and files dynamically generated by a catalog server for that appliance. As necessary, the appliance requests and installs any updated packages and files from a component server. Each package includes self-installing instructions and is authenticated and decrypted prior to installation. Each file is received over a secure connection and is installed per instructions stored in a file information subdirectory at the networks operations center.

An embodiment of the present invention provides a system and a method for providing a framework for network appliance management in a distributed computing environment. A status report periodically received from each of a plurality of network appliances is recorded. Each status report contains health and status information and application-specific data for each network appliance. Configuration settings for each network appliance progressively assembled concurrent to providing installable components are maintained. A catalog listing currently installable components for each network appliance based on the configuration settings is dynamically provided.

A further embodiment provides a system and method for autonomously managing a network appliance deployed

within a distributed computing environment. An internal catalog of components installed on one such network appliance is maintained, identified by component and version. A status report containing health and status information and application-specific data is periodically provided for the one such network appliance. A catalog of currently installable components dynamically generated for the one such network appliance is obtained. Non-current components are determined by comparing the components and versions listed in the obtained catalog against the internal catalog.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing a system for providing a framework for network appliance configuration management.

FIG. 2 is a block diagram showing the software modules of the individual servers comprising the system of FIG. 1.

FIG. 3 is a block diagram showing the software modules of an exemplary network appliance of FIG. 1.

FIG. 4 is a process flow diagram showing remote network appliance management, as performed by the system of FIG. 1.

FIG. 5 is a data structure diagram showing a package maintained by the component server of FIG. 2.

FIG. 6 is a flow diagram showing a method for providing a framework for network appliance management, in accordance with the present invention.

FIG. 7 is a flow diagram showing the process performed by the network operations center of FIG. 2.

FIG. 8 is a flow diagram showing the process performed by the catalog server of FIG. 2.

FIG. 9 is a flow diagram showing the process performed by the component server of FIG. 2.

FIGS. 10A and 10B are flow diagrams showing the process performed by the network appliance of FIG. 3.

FIG. 11 is a flow diagram showing the routine for downloading a package for use in the process of FIGS. 10A and 10B.

FIG. 12 is a flow diagram showing the routine for downloading a file for use in the method of FIGS. 10A and 10B.

FIG. 13 is a flow diagram showing the daemon process performed by the network operations center of FIG. 2.

DETAILED DESCRIPTION

FIG. 1 is a network diagram 10 showing a system for providing a framework for network appliance management in a distributed computing environment, in accordance with the present invention. The distributed computing environment is preferably TCP/IP compliant. A plurality of individual network appliances (or simply "appliances") 11a-c are interconnected via an intranetwork 13. Each of the appliances 11a-c is autonomously managed and provides specified functionality, such as electronic mail (email) anti-

virus scanning, content filtering, packet routing, or file, Web, or print service. Other forms of appliance services are feasible, as would be recognized by one skilled in the art.

In addition to providing the specified functionality, the various appliances 11a-c are autonomously self-configured and self-managed, as further described below with reference to FIG. 3. Each appliance 11a-c periodically generates reports on status and health and provides application-specific data, known as "SecureBeats," to a centralized network operations center (NOC) 12. Each appliance 11a-c then obtains a catalog from a catalog server 15 operating on the network operations center 12. As necessary, packages and files are obtained from a component server 16, or alternatively, local component server 18. Packages and files are updated whenever the downloaded catalog indicates that a currently installed package or file is out of date.

Each appliance 11a-c is interconnected via an intranetwork 13 which is, in turn, interconnected to an internetwork 20, including the Internet, via a firewall 21 and border router 22. The local component server 18 is also interconnected via the intranetwork 13 and shares the same network domain with the appliances 11a-c. The network operations center 12 and component server 16 are external to the intranetwork 13 and are only accessible as remote hosts via the internetwork 20. Accordingly, the reporting and catalog functions are transacted with each appliance 11a-c in a secure session, preferably using the Secure Hypertext Transport Protocol (HTTPS). Furthermore, as further described below with reference to FIG. 2, the file download function must also be transacted in a secure session. Other network configurations, topologies and arrangements of clients and servers are possible, as would be recognized by one skilled in the art.

Each appliance 11a-c maintains an internal catalog listing the packages and files currently installed. Immediately upon being interconnected to the intranetwork 13, each appliance 11a-c is remotely configured using a Web browser-based configuration, which installs the various packages and files providing the specific functionality of the appliance, such as described in commonly-assigned related U.S. patent application Ser. No. 10/057,709, filed Jan. 25, 2002, pending, the disclosure of which is incorporated by reference.

On a regular periodic basis, each appliance 11a-c awakens and contacts the network operations center (NOC) 12 to upload the status report. Alternatively, the network operations center 12 can broadcast a "ping" query message to all appliances 11a-c to wake up each appliance 11a-c and trigger a status report upload. Next, each appliance 11a-c contacts the catalog server 15 to retrieve a copy of a catalog of the most-up-to-date packages and files currently available. The catalog server 15 executes as part of the network operations center 12 and maintains a catalog database 14. The catalog server 15 dynamically generates a catalog for each requesting appliance 11a-c based on the type and configuration of appliance.

Upon receiving the catalog from the catalog server 15, the appliance 11a-c determines whether updates to the configuration or installed applications are necessary. Updates are effected by downloaded components which include packages and files. If an update is required, the appliance 11a-c requests and "pulls" the identified packages and files from the component server 16 or, alternatively, the local component server 18, for download. The component server 16 and local component server 18 maintain databases, component database 17 and local component database 19, respectively, in which the most-up-to-date packages and files are stored. The appliance 11a-c downloads the required packages and files for subsequent installation.

In a further embodiment, the functionality of the network operations center **12** and component server **16** can be combined into a single server (not shown) or implemented on separate systems for each of the network operations center **12**, catalog server **15**, and component server **16**. The use of separate servers for publishing the catalog and providing component downloads of packages and files allows finer-grained distributed processing of network appliance configuration and management.

Individual packages and files are optionally staged for download by a local component server **18** interconnected via the intranetwork **13**. The close proximity of the local server **18** to the appliances **11a-c** allows for faster and more convenience component downloads and avoids bandwidth congestion at the border router **22**.

The individual computer systems, including servers and clients, are general purpose, programmed digital computing devices consisting of a central processing unit (CPU), random access memory (RAM), non-volatile secondary storage, such as a hard drive or CD ROM drive, network interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs and data, are loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage.

FIG. **2** is a block diagram showing the software modules **30** of the individual servers comprising the system **10** of FIG. **1**. The servers include the network operations center **12**, catalog server **15** and component server **16**, plus local component server **18**.

The network operations center **12** includes two modules: status monitor **31** and status daemon **32**. The catalog server **15** executes as part of the network operations center **12**. The status monitor **31** receives the periodic status reports from the individual network appliances **11a-c** (shown in FIG. **1**), as further described below with reference to FIG. **7**. Each status report is recorded and registered in an appliance status table **33**, which notes the appliance user identifier (UID) and time of each report.

The status daemon **32** executes as an independent process that periodically awakens and examines the appliance status table **33** to determine whether any of the appliances **11a-c** have failed to report, as further described below with reference to FIG. **13**. As necessary, an alert is generated to inform an administrator of a potentially faulty appliance.

The catalog server **15** includes four modules: validation **34**, catalog engine **35**, database **36**, and crypto **37**. The validation module **34** validates catalog requests received from individual appliances **11a-c**. In the described embodiment, each appliance **11a-c** sends a user identifier (UID) as part of each catalog request, which is used to validate the identity of the requesting appliance.

The catalog engine **35** dynamically generates catalogs **38** listing the most-up-to-date packages and files for download on an individual appliance basis. In the described embodiment, the catalogs **38** are generated in the Extensible Markup Language (XML), although any other form of catalog description could also be used. The catalog engine **35** refers to the appliance status table **33** to determine the current configuration of each appliance.

The database module **36** interfaces to the main database **14** to access the catalogs **38** maintained therein. In the described embodiment, the main database **14** is a structured query language (SQL) based database. The catalog information is stored as structured records indexed by user identifiers.

The crypto module **37** provides asymmetric (public key) and symmetric encryption. Both forms of cryptography are needed to transact a secure session with each appliance **11a-c**. As well, the network operations center **12** uses the crypto module **37** to digitally sign and encrypt packages that are staged in the component database **17** and local component database **19** (both shown in FIG. **1**).

The component server **16** includes four modules: validation **39**, component download **40**, database **41** and **42**. The validation module **40** validates component requests received from individual appliances **11a-c**. In the described embodiment, each appliance **11a-c** sends a user identifier (UID) as part of each component request, which is used to validate the identity of the requesting appliance.

The component download module **40** downloads requested packages **43** and files **44** to validated network appliances **11a-c**. The component download module **40** records the names and versions of applications installed on each network appliance **11a-c** by maintaining a set of configuration settings (not shown) for each network appliance **11a-c** progressively assembled concurrent to the downloading of each requested package **43** and file **44**. Accordingly, the persistent configured state and applications suite installed on a network appliance **11a-c** could be completely restored by the component server **16**, should the set of installed applications on any given network appliance **11a-c** become corrupt or rendered otherwise unusable through a catastrophic crash or service termination.

As further described below with reference to FIG. **4**, each package **43** contains an encrypted self-contained set of installable software digitally signed by the network operations center **12**. Packages can be downloaded in now-secure sessions. Files **45** are not signed or encrypted and must be downloaded in secure sessions. The installation location on a given appliance **11a-c** is determined by the instructions encoded in a file-information subdirectory on the network operations center **12**.

The database module **41** interfaces to the component database **17** to access the packages **43** and files **44** maintained therein. In the described embodiment, the component database **17** is a structured query language (SQL) based database.

The crypto module **42** provides asymmetric (public key) and symmetric encryption. Both forms of cryptography are needed to transact a secure session with each appliance **11a-c**.

The functionality of the local component server **18** and local component database **19** is substantially identical to that of the component server **16** and component database **17**. The only distinction between the two component servers is the location of each within the system **10** of FIG. **1**. The local component server **18** effectively functions as a proxy component server by staging components for convenient download by locally proximate network appliances.

In addition, the functionality of the network operations center **12**, catalog **15** and component server **16** could be combined into a single integrated server or provided as separate systems deployed in various locations and combinations throughout the intranetwork **13** and internetwork **20**, as would be recognized by one skilled in the art.

FIG. **3** is a block diagram showing software modules **50** of an exemplary network appliance **11a** of FIG. **1**. Application-specific logic has been omitted for clarity. As pertains to autonomous configuration and management, each network appliance **11a** includes four modules: catalog checker **51**, crypto **52**, installer **53**, and status daemon **54**. The catalog checker **51** requests and examines a catalog returned

from the catalog server **15** (shown in FIG. 1) to determine whether software updates are required. Each downloaded catalog, `catalog.new`, is checked against an internal catalog **55**, `catalog.cur`. The internal catalog **55** lists the installed applications **56** currently used by the appliance **11a**. Required packages **57** and files **58** are downloaded or “pulled” from the component server **16**. The installed applications **56** include both the functional programs implemented on each network appliance **11a-c** to perform the application-specific logic for a given function, as well as operating system and support software, including the software modules **50**. Accordingly, the autonomous configuration and self-management of each network appliance **11a-c** can enable a vendor to provide a complete service model whereby updates and device recovery is handled automatically and without end-user intervention.

The crypto module **52** provides asymmetric (public key) and symmetric encryption. Both forms of cryptography are needed to transact a secure session with the network operations center **12** and component server **16**. Public key encryption is also used to authenticate and decrypt downloaded packages **57**.

The installer **53** installs downloaded packages **57** and files **58**. Each individual package **57** includes a complete setup program, as further described below with reference to FIG. 5. Each file **58** must be installed in a location identified in a corresponding file information subdirectory, `fileinfo`, on the network operations center **12**.

Finally, the status daemon **54** periodically awakens and sends a report of the health and status of the network appliance **11a** to the network operations center **12**. The status report identifies the reporting appliance **11a** and provides machine-specific data, including the load on the processor, available disk space application-specific information, such as the number of emails passing through the device. The status report is referred to as a “SecureBeat.”

Each software module of the individual servers and exemplary appliance **11a** is a computer program, procedure or module written as source code in a conventional programming language, such as the C++ programming language, and is presented for execution by the CPU as object or byte code, as is known in the art. The various implementations of the source code and object and byte codes can be held on a computer-readable storage medium or embodied on a transmission medium in a carrier wave. The individual servers and exemplary appliance **11a** operate in accordance with a sequence of process steps, as further described beginning below with reference to FIG. 6.

FIG. 4 is a process flow diagram showing remote network appliance management, as performed by the system of FIG. 1. Each network appliance **11a-c** is autonomously managed. Management requires two mandatory phases, status reporting (step **61-63**) and catalog examination (step **64-67**), and two optional phases, package downloading (steps **68-71**) and file downloading (steps **72-75**).

During the status reporting phase, the appliance **11a** requests a secure session, preferably using HTTPS (step **61**). Upon the creation of a secure session (step **62**), the appliance **11a** sends a status report to the network operations center **12** (step **63**). The status report is then logged by the network operations center **12**.

During the catalog examination phase, the appliance **11a** requests a secure session, preferably using HTTPS (step **64**). Upon the creation of a secure session (step **65**), the appliance **11a** posts a user identifier (UID) (step **66**). The catalog server **15** validates the identity of the requesting appliance **11a** and, if valid, archives that the particular appliance **11a**

has connected at the current time in the appliance status table **33** (shown in FIG. 2). The catalog server **15** then downloads the catalog to the requesting appliance **11a** (step **67**).

The package downloading and file downloading phases are performed when the appliance **11a** determines that an installed application **56** (shown in FIG. 3) is out of date. During the package downloading phase, the appliance **11a** requests a non-secure session (step **68**). Upon the creation of a non-secure session (step **69**), the appliance **11a** requests the necessary packages (step **70**). The component server **16** validates the identity of the requesting appliance **11a** by examining the package request. Each package request includes a user identifier (UID) and uniform resource locator (URL) indicating the location of the required package. The package is then downloaded to the requesting appliance **11a** (step **71**) for installation.

After receiving each package, the requesting appliance **11a** clarifies that the package has been signed with the appropriate private key for the network operations center **12** using public key authentication. The package downloading phase (steps **70-71**) is repeated until all required packages have been downloaded.

During the file downloading phase, the appliance **11a** requests a secure session, preferably using HTTPS (step **72**). Upon establishing a secure session (step **73**), the appliance **11a** requests the necessary files (step **74**). The component server **16** validates the identity of the requesting appliance **11a** by examining the file request. Each file request includes a user identifier (UID) and uniform resource locator (URL) indicating the location of the required file. The file is then downloaded to the requesting appliance **11a** (step **75**) for installation.

After receiving each file, the requesting appliance **11a** installs the downloaded file based on installation instructions found in the file information subdirectory on the network operations center **12**. The file downloading phase (steps **74-75**) is repeated until all required files have been downloaded.

In the described embodiment, the package downloading (steps **68-71**) and file downloading (steps **72-75**) phases must occur in sequential order. Individual packages can contain placeholder files that must be overwritten by appliance-specific files following package installation. To allow such appliance-specific dependencies, the packages must generally be installed first.

FIG. 5 is a data structure diagram **80** showing a package **81** maintained by the component server **16** of FIG. 1. Each package **81** includes a digital signature **82** that authenticates the package as having originated with the network operations center **12**. Only those packages **81** containing a properly-authenticated digital signature are installed by the individual network appliances **11a-c** (shown in FIG. 1). Each package **81** also includes an executable program **83**, `install.exe`, which is executed by the appliance **11a-c** to effect the installation of the package **81**. Finally, each package contains the individual files to be installed **84** by the executable program **83**.

Unlike packages, each file does not contain specific instructions for installation. Instead, the installing appliance **11a-c** looks up the appropriate instructions in a specific file information subdirectory, called `fileinfo`, located on the network operations center **12**. In the described embodiment, the subdirectory is located under `$USER/SecureBeat/Upload/Application-Name`, where `$USER` is a root directory and `Application-Name` refers to an installed application **56** (shown in FIG. 3).

For example, to update a virus screening application, the subdirectory would be "\$USER/SecureBeat/Upload/V-Screen" and the configuration file would be vscreen.upload.comp. The contents of a sample configuration file are as follows:

```
<UPLOAD DIR="VSCREEN" SERVER="BWSH">
```

The tag UPLOAD indicates the file is located in the subdirectory called "VSCREEN" and is to be retrieved from the server "BWSH."

FIG. 6 is a flow diagram 100 showing a method for providing a framework for network appliance management, in accordance with the present invention. The individual components, including network operations center 12, catalog server 15, component server 16 and individual network appliances 11a-c, execute independently. Each of the components must be initialized and started (blocks 101-104) prior to appliance management. Upon respective initialization and starting, each component proceeds independently, as further described below with reference to FIGS. 7-10.

FIG. 7 is a flow diagram 110 showing the process performed by the network operations center 12 of FIG. 2. The network operations center 12 executes an iterative processing loop (blocks 111-115). During each iteration (block 111), a secure session is established with a requesting appliance (block 112). Upon establishing a secure session, a status report is received from each appliance (block 113), after which the secure session is closed (block 114). During the secure session, the appliance reports the health and status of the machine. Processing continues (block 115) until the process is terminated or halted.

FIG. 8 is a flow diagram 120 showing the process performed by the catalog server 15 of FIG. 2. The catalog server 15 executes an iterative processing loop (blocks 121-129). During each iteration (block 121), a secure session is established with a requesting appliance (block 122). Upon establishing a secure session, the catalog server 15 receives the user identification (UID) of the requesting appliance (block 123). If the user identification is not valid (block 124), an error condition is generated (block 125) and the administrator is notified. Otherwise, the connection time of the requesting appliance is archived (block 126) in the appliance status table 34 (shown in FIG. 2). A catalog is dynamically generated and sent to the requesting client (block 127). In the described embodiment, the catalog is generated as an XML document, although any other type of catalog description format could be used. The secure session is then closed (block 128). Processing continues (block 129) until the process is terminated.

FIG. 9 is a flow diagram 140 showing the process performed by the component server 16 of FIG. 2. The component server 16 executes an iterative processing loop (blocks 141-149). During each iteration (block 141), the component server 16 first establishes a secure session with the requesting appliance (block 142). The requesting appliance sends a request for an individual component to download which includes a user identifier (UID) and a URL indicating the location of the component to be downloaded. The request is received (block 143) and validated. If the user identification is not valid (block 144), an error condition is generated (block 145) and the administrator is notified. Otherwise, the requested component is downloaded to the requesting appliance (block 146) and the database is flagged to indicate that the downloaded component was sent to the requesting appliance (block 147). The component server 16 then closes the present session (block 148). Note a secure session is required for downloading files while a non-secure

session is used when downloading packages. Processing continues (block 149) until the process is terminated or halted.

FIGS. 10A and 10B are flow diagrams 160 showing the process performed by the network appliance 11a of FIG. 3. Each network appliance 11a-c (shown in FIG. 1) periodically awakens, sends a status report, receives a catalog, and downloads any required packages and files. In addition, each network appliance 11a-c executes any initial plug-ins and post-plug-ins prior to and following the reporting and updating phase (shown in FIG. 4).

Thus, each network appliance 11a-c periodically awakens (block 161). In the described embodiment, each appliance 11a-c awakens once every 15 minutes, nine seconds. Any installed initial plug-ins are executed (block 162). By way of example, initial plug-ins include executables which monitor daemon processes, which must always be running. An initialization plug-in called "VScreen.init" executes as a watchdog process to determine if the daemon process is still running. The daemon process is restarted as necessary. As well, individual status reports are generated by the initial plug-ins, which must be executed prior to the reporting phase.

After executing any initial plug-ins, a secure session is established with the network operations center 12 (block 163) and the status report is sent (block 164). The secure session is then closed (block 165). A secure session is then established with the catalog server 16 (block 166) and the user identifier (UID) is sent (block 167). Upon validation by the catalog server 15, a dynamically-generated catalog is received from the catalog server 15 (block 168). The secure session is then closed (block 169).

Upon receiving the catalog from the catalog server 15, the catalog is examined (block 170). Each catalog includes a list of component names and versions, a tag indicating the server at which to locate and obtain the component, and the type of component, that is, package or file. If components are required (block 172), packages are first iteratively downloaded (blocks 172-174) followed by files (blocks 175-177). For each package (block 172), the package is downloaded (block 173), as further described below with reference to FIG. 11. Similarly, for each file (block 175), the file is downloaded (block 176), as further described below with reference to FIG. 12.

Upon completion of the downloading of each required package and file, any post plug-ins are executed (block 178). Finally, the network appliance returns to a sleep mode (block 179). Processing continues until the process is terminated or halted.

FIG. 11 is a flow diagram 180 showing the routine for downloading a package for use in the process of FIGS. 10A and 10B. The purpose of this routine is to connect to the component server 16 and retrieve any required packages for installation by an appliance 11a.

Thus, a non-secure session is first established with the component server 16 (block 181). A Uniform Resource Locator (URL), including the user identifier (UID), package name and version, are sent to the component server 16 (block 182). Upon being credentialed by the component server, the requested package is received (block 183) and the non-secure session is closed (block 184).

Each individual package 57 is authenticated and encrypted by the network operations center 12 prior to being staged in the component database 17 of the component server 16. Accordingly, the downloading appliance 11a first verifies that the package was digitally signed with the private key for the network operations center 12 (block 185),

11

after which the package is installed (block 186), following the instructions stored therein. The routine then returns.

FIG. 12 is a flow diagram 190 showing the routine for downloading a file 190 for use in the process of FIGS. 10A and 10B. The purpose of this routine is to connect to the component server 16 and retrieve any required files for installation by an appliance 11a. Each network appliance 11a-c connects to and downloads required files for installation per the instructions included in a file information subdirectory on the network operations center 12.

Thus, the network appliance 11a establishes a secure session with the component server 16 (block 191) and sends a uniform resource locator (URL), including user identifier (UID), file name and version, to the component server 16 (block 192). Upon being credentialed by the component server 16, the requested file is received (block 193), and the secure session is closed (block 194). The file is installed based on the information stored in the file information subdirectory (block 195). The routine then returns.

FIG. 13 is a flow diagram 200 showing the daemon process performed by the network operations center 12 of FIG. 2. The daemon process periodically awakens (block 201) and iteratively checks the status of each configured network appliance 11a-c (shown in FIG. 1) managed by the network operations center 12. During each iteration (block 202), the network operations center 12 determines whether a status report has been received from each of the appliances 11a-c since the last reporting cycle (block 203) by examining the appliance status table 34 (shown in FIG. 2). If a report has not been received (block 204), an error is generated (step 205) and the administrator is notified. Processing continues with each successive appliance (block 206), after which the daemon process returns to sleep (block 207).

While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A system for providing a framework for network appliance management in a distributed computing environment, comprising:

an appliance status table recording a status report periodically received from a status daemon autonomously operating on each of a plurality of network appliances, each status report containing health and status information and application-specific data pertaining to autonomous configuration and management of each network appliance; and

a catalog server maintaining configuration settings for each network appliance progressively assembled concurrent to providing installable components and dynamically providing a catalog listing currently installable components for being installed on each network appliance based on the configuration settings independently received from the network appliance;

wherein each network appliance, prior to sending the status report, executes at least one initial plug-in; and, after installing the installable components, executes at least one post-plug-in;

wherein the at least one initial plug-in monitors the status daemon to determine if the status daemon is running, and restart the status daemon if it is determined that the status daemon is not running;

wherein the catalog further includes installable component names, installable component versions, a tag indicating a component server at which to locate and obtain

12

each installable component, and a type indicator indicating whether each installable component is a package or a file;

wherein a network operations center establishes a secure session with each network appliance utilizing Secure Hypertext Transfer Protocol (HTTPS);

wherein the appliance status table further records a user identifier associated with one of the network appliances from which the status report is received and a time the status report is received.

2. A system according to claim 1,

wherein the network operations center installs an initial set of installable components on each network appliance during a bootstrap configuration.

3. A system according to claim 1, wherein the currently installable components comprise at least one self-installable package, and the component server supplies the at least one package for installation responsive to a request from one such network appliance.

4. A system according to claim 3, further comprising: a crypto module digitally signing the at least one package for the network operations center prior to being supplied for installation.

5. A system according to claim 3, further comprising: a crypto module encrypting the at least one package prior to being supplied for installation.

6. A system according to claim 1, wherein the installable components comprise at least one file, and the component server supplies the at least one file responsive to a request from one such network appliance.

7. A system according to claim 6, wherein the component server establishes the secure session prior to the at least one file being supplied for installation.

8. A system according to claim 6, further comprising: a file information subdirectory specifying installation instructions for the at least one file in a pre-determined entry prior to the at least one file being supplied for installation.

9. A system according to claim 1, further comprising: a proxy component server staging the currently installable components for retrieval in a separate components database.

10. A system according to claim 1, wherein the distributed computing environment is TCP/IP-compliant.

11. A system according to claim 1, wherein the status report contains machine-specific data including a load on a processor and available disk space associated with each network appliance, and the application-specific data includes a number of e-mails passing through each of a plurality of network devices.

12. A system according to claim 1, wherein the installable components for being installed on each network appliance are installed on each network appliance according to a location identified in a corresponding file information subdirectory of the network operations center.

13. A method for providing a framework for network appliance management in a distributed computing environment, comprising:

recording a status report periodically received from a status daemon autonomously operating on each of a plurality of network appliances, each status report containing health and status information and application-specific data pertaining to autonomous configuration and management of each network appliance;

maintaining configuration settings for each network appliance progressively assembled concurrent to providing installable components; and

13

dynamically providing a catalog listing currently installable components for being installed on each network appliance based on the configuration settings independently received from the network appliance;

wherein each network appliance, prior to sending the status report, executes at least one initial plug-in; and, after installing the installable components, executes at least one post-plug-in;

wherein the at least one initial plug-in monitors the status daemon to determine if the status daemon is running, and restart the status daemon if it is determined that the status daemon is not running;

wherein the catalog further includes installable component names, installable component versions, a tag indicating a component server at which to locate and obtain each installable component, and a type indicator indicating whether each installable component is a package or a file;

wherein a secure session is established with each network appliance utilizing Secure Hypertext Transfer Protocol (HTTPS);

wherein a user identifier associated with one of the network appliances from which the status report is received and a time the status report is received are recorded.

14. A method according to claim 13, further comprising: installing an initial set of installable components on each network appliance during a bootstrap configuration.

15. A method according to claim 13, wherein the currently installable components comprise at least one self-installable package, further comprising:

- supplying the at least one package for installation responsive to a request from one such network appliance.

16. A method according to claim 15, further comprising: digitally signing the at least one package prior to being supplied for installation.

17. A method according to claim 15, further comprising: encrypting the at least one package prior to being supplied for installation.

18. A method according to claim 13, wherein the installable components comprise at least one file, further comprising:

- supplying the at least one file responsive to a request from one such network appliance.

19. A method according to claim 18, further comprising establishing the secure session prior to the at least one file being supplied for installation.

20. A method according to claim 18, further comprising: specifying installation instructions for the at least one file in a pre-determined entry prior to the at least one file being supplied for installation.

21. A method according to claim 13, further comprising: staging the currently installable components for retrieval in a separate components database.

22. A method according to claim 13, wherein the distributed computing environment is TCP/IP-compliant.

23. A computer-readable storage medium holding code for performing the method according to claims 13, 14, 15, 16, 17, 18, 19, 20, 21, or 22.

24. A system for autonomously managing a network appliance deployed within a distributed computing environment, comprising:

- an internal catalog of components installed on one such network appliance identified by component and version; and
- a status daemon operating autonomously on the one such network appliance and periodically providing a status

14

report containing health and status information and application-specific data pertaining to autonomous configuration and management of the one such network appliance; and

a catalog checker obtaining a catalog of currently installable components dynamically generated for the one such network appliance based on the status report independently received from the one such network appliance and determining non-current components by comparing the components and versions listed in the obtained catalog against the internal catalog;

wherein each network appliance, prior to sending the status report, executes at least one initial plug-in; and, after installing the installable components, executes at least one post-plug-in;

wherein the at least one initial plug-in monitors the status daemon to determine if the status daemon is running, and restart the status daemon if it is determined that the status daemon is not running;

wherein the catalog further includes a tag indicating a component server at which to locate and obtain each installable component, and a type indicator indicating whether each installable component is a package or a file;

wherein a network operations center negotiates a secure session with the one such network appliance utilizing Secure Hypertext Transfer Protocol (HTTPS);

wherein an appliance status table records a user identifier associated with the one such network appliance from which the status report is received and a time the status report is received.

25. A system according to claim 24, wherein the components comprise at least one self-installable package, further comprising:

- an installer obtaining the at least one self-installable package and installing the at least one self-installable package per instructions encoded therein.

26. A system according to claim 25, wherein the components further comprise at least one file dependent on the at least one self-installable package, further comprising:

- an installer obtaining the at least one file subsequent to installing the at least one self-installable package and installing the at least one self-installable package per instructions stored in a pre-determined entry.

27. A system according to claim 25, wherein the component server negotiates a non-secure session prior to obtaining the at least one self-installable package.

28. A system according to claim 25, further comprising: a crypto module at least one of authenticating and decrypting the at least one self-installable package prior to installing the at least one self-installable package.

29. A system according to claim 25, wherein the instructions comprise an executable installation program plus one or more files to be installed.

30. A system according to claim 25, wherein the components further comprise at least one file, further comprising:

- an installer obtaining the at least one file and installing the at least one self-installable package per instructions stored in a predetermined entry.

31. A system according to claim 30, wherein the component server negotiates the secure session prior to obtaining the at least one self-installable package.

32. A system according to claim 30, wherein the pre-determined entry comprise a file information subdirectory identifying installation instructions.

15

33. A system according to claim 25, wherein at least one such network appliance performs one of electronic mail anti-virus scanning, content filtering, packet routing, and file, Web and print servicing.

34. A system according to claim 25, wherein the distributed computing environment is TCP/IP-compliant.

35. A method for autonomously managing a network appliance deployed within a distributed computing environment, comprising:

maintaining an internal catalog of components installed on one such network appliance identified by component and version;

periodically providing a status report containing health and status information and application-specific data pertaining to autonomous configuration and management of the one such network appliance and received from a status daemon autonomously operating on for the one such network appliance,

obtaining a catalog of currently installable components dynamically generated for the one such network appliance based on the status report independently received from the one such network appliance; and

determining non-current components by comparing the components and versions listed in the obtained catalog against the internal catalog;

wherein each network appliance, prior to sending the status report, executes at least one initial plug-in; and, after installing the installable components, executes at least one post-plug-in;

wherein the at least one initial plug-in monitors the status daemon to determine if the status daemon is running, and restart the status daemon if it is determined that the status daemon is not running;

wherein the catalog further includes a tag indicating a component server at which to locate and obtain each installable component, and a type indicator indicating whether each installable component is a package or a file;

wherein a secure session is negotiated with the one such network appliance utilizing Secure Hypertext Transfer Protocol (HTTPS);

wherein a user identifier associated with the one such network appliance from which the status report is received and a time the status report is received are recorded.

36. A method according to claim 35, further comprising: broadcasting a query message to each such network appliance to trigger a status report.

16

37. A method according to claim 35, wherein the components comprise at least one self-installable package, further comprising:

obtaining the at least one self-installable package; and installing the at least one self-installable package per instructions encoded therein.

38. A method according to claim 37, wherein the components further comprise at least one file dependent on the at least one self-installable package, further comprising:

obtaining the at least one file subsequent to installing the at least one self-installable package; and installing the at least one self-installable package per instructions stored in a pre-determined entry.

39. A method according to claim 37, further comprising; negotiating a non-secure session prior to obtaining the at least one self-installable package.

40. A method according to claim 37, further comprising: at least one of authenticating and decrypting the at least one self-installable package prior to installing the at least one self-installable package.

41. A method according to claim 37, wherein the instructions comprise an executable installation program plus one or more files to be installed.

42. A method according to claim 35, wherein the components further comprise at least one file, further comprising:

obtaining the at least one file; and installing the at least one self-installable package per instructions stored in a pre-determined entry.

43. A method according to claim 42, further comprising: negotiating the secure session prior to obtaining the at least one self-installable package.

44. A method according to claim 42, wherein the pre-determined entry comprise a file information subdirectory identifying installation instructions.

45. A method according to claim 35, wherein at least one such network appliance performs one of electronic mail anti-virus scanning, content filtering, packet routing, and file, Web and print servicing.

46. A method according to claim 35, wherein the distributed computing environment is TCP/IP-compliant.

47. A computer-readable storage medium holding code for performing the method according to claims 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, or 46.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,089,259 B1
APPLICATION NO. : 10/056702
DATED : August 8, 2006
INVENTOR(S) : Kouznetsov et al.

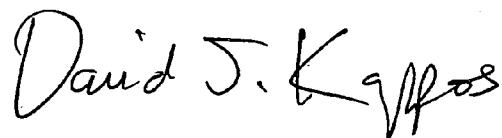
Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

col. 15, line 18 replace "appliance," with --appliance;--;
col. 16, line 14 replace "comprising;" with --comprising:--.

Signed and Sealed this

Ninth Day of February, 2010

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive style with a large, stylized 'D' and 'K'.

David J. Kappos
Director of the United States Patent and Trademark Office

NETFLIX, INC. EXHIBIT 1002

APPENDIX B-8



(19) **United States**

(12) **Patent Application Publication**

Kao et al.

(10) **Pub. No.: US 2007/0192652 A1**

(43) **Pub. Date: Aug. 16, 2007**

(54) **RESTRICTING DEVICES UTILIZING A DEVICE-TO-SERVER HEARTBEAT**

Publication Classification

(75) Inventors: **Sandy Kao**, Austin, TX (US); **Rodrigo J. Pastrana**, Delray Beach, FL (US)

(51) **Int. Cl.**
G06F 11/00 (2006.01)
(52) **U.S. Cl.** **714/4**

Correspondence Address:
PATENTS ON DEMAND, P.A.
4581 WESTON ROAD
SUITE 345
WESTON, FL 33331 (US)

(57) **ABSTRACT**

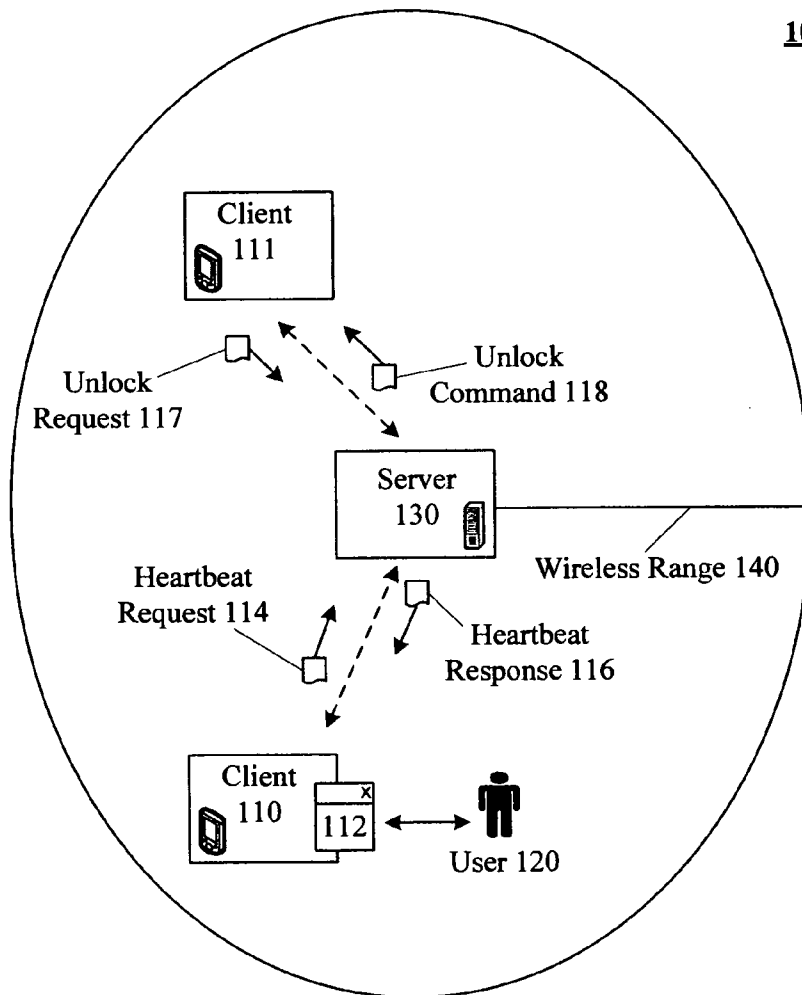
A method of automatically locking a client can include a step of a client automatically establishing a heartbeat interval. A determination can be automatically made regarding whether a proper server response is received within the heartbeat interval. When no proper response is received, the client can be automatically placed in a locked state. All client functions accessible by a user other than those functions relating to unlocking the client can be disabled while the client is in the locked state. A remotely located server can unlock the client by conveying an unlock message to the client.

(73) Assignee: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY

(21) Appl. No.: **11/354,477**

(22) Filed: **Feb. 14, 2006**

100



100

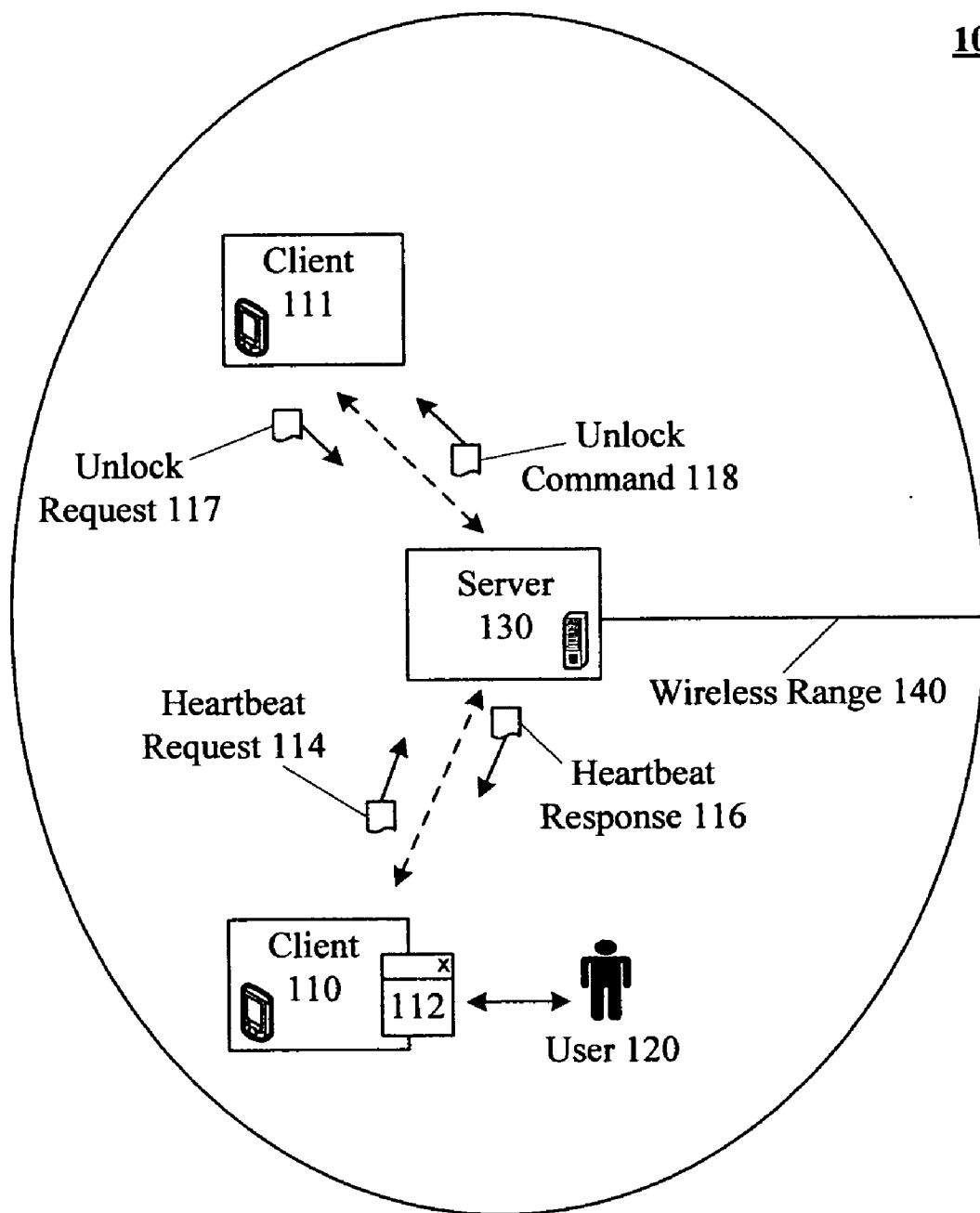


FIG. 1

200

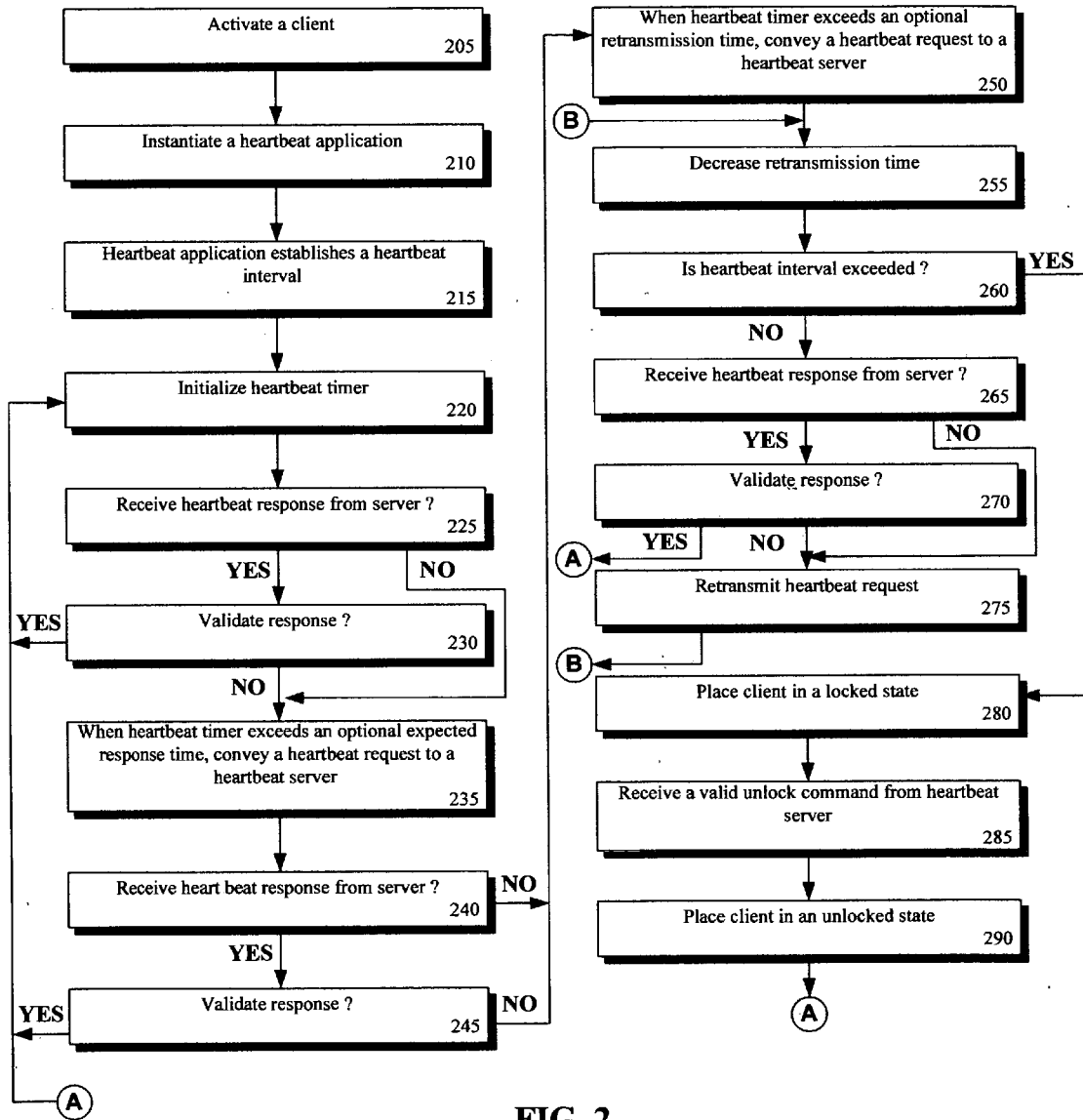


FIG. 2

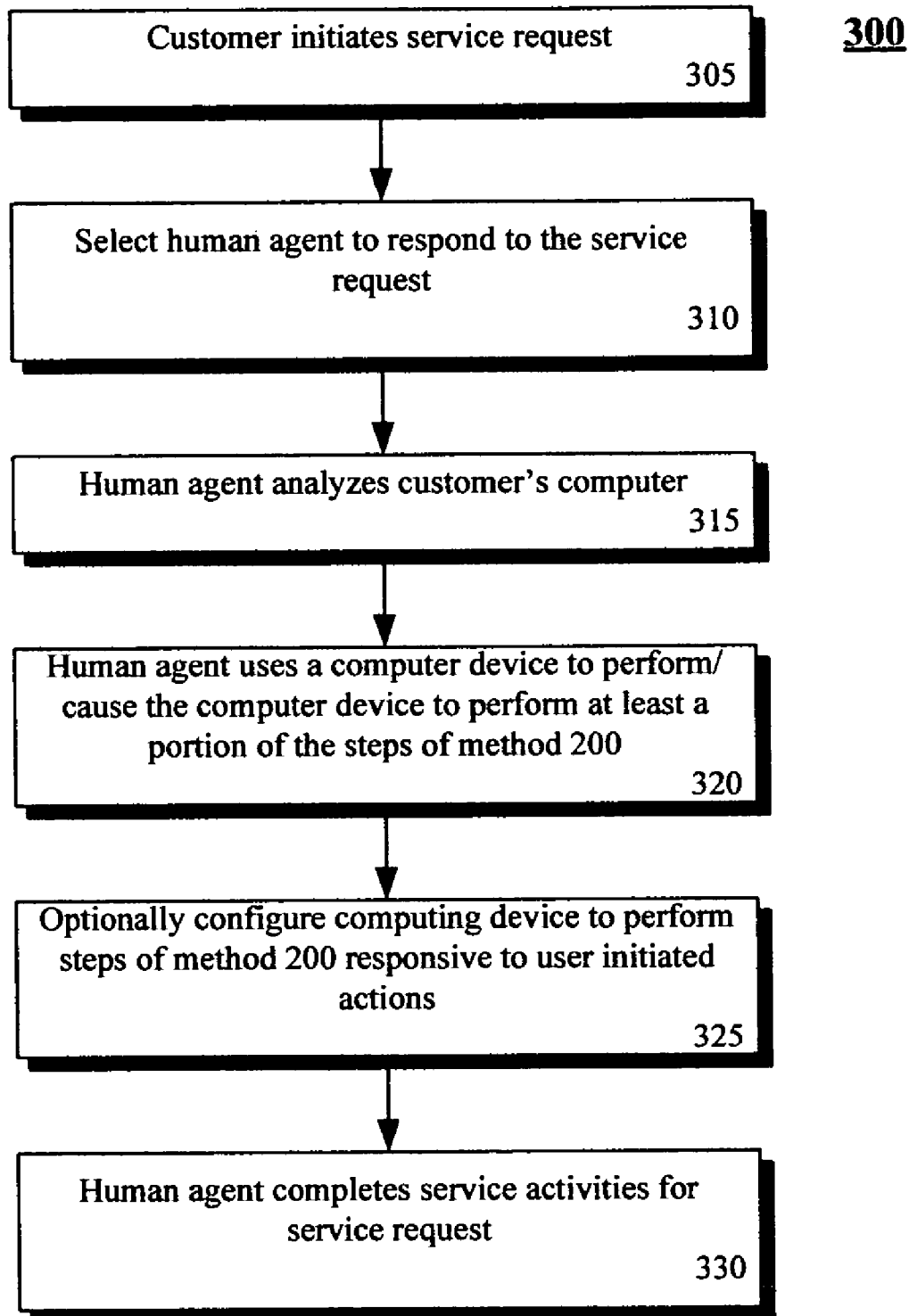


FIG. 3

RESTRICTING DEVICES UTILIZING A DEVICE-TO-SERVER HEARTBEAT

BACKGROUND

[0001] 1. Field of the Invention

[0002] The present invention relates to the field of computer security, and, more particularly, to restricting computing devices utilizing a device-to-server heartbeat.

[0003] 2. Description of the Related Art

[0004] Businesses are increasingly relying upon computing devices to perform business tasks. For example, in addition to desktop computers, businesses often provide mobile telephones, personal data assistants (PDAs), bar code scanners, tablet computing devices, notebooks, kiosks, and other devices for use by customers and employees. Individual ones of these devices are often shared between employees and/or customers. These devices are often portable devices that are optimally placed in locations of high availability.

[0005] The cost and availability of the devices result in a high risk of theft. Theft of the devices usually has one of three different goals: (1) to personally use a stolen device, (2) to resell the stolen device, and (3) to extract sensitive information from the stolen device. Conventional techniques to prevent device theft have significant shortcomings.

[0006] For example, it is common to physically constrain a device to a location using a chain/lock combination. This solution can greatly restrict the placement and mobility of a device, which decreases its usefulness in a business setting. Also, physical security precautions can require active measures be taken by employee users, which are often ignored or forgotten.

[0007] Other security solutions attempt to restrict, locate, or disable a device after a theft has been detected. For example, software can be loaded and hidden on the device that causes the device to broadcast a beacon or to take a restrictive action responsive to a command received via the Internet. These post theft solutions are flawed since each requires the stolen device to be able to receive commands via a network. Conventional software-based theft deterrents are also able to be removed from a device by a device user. For these reasons, conventional anti-theft solutions are inadequate to prevent device thefts. That is, even when conventional anti-theft solutions are implemented, the goals of most device thieves can still be achieved.

SUMMARY OF THE INVENTION

[0008] The present invention executes a daemon or application upon a computing device that generates a heartbeat for the device. The heartbeat is associated with a timer and a timed operation interval, referred to as a heartbeat interval. The device can be used in a stand-alone as well as in a networked fashion for the heartbeat interval. Before the end of the interval, the device requires a heartbeat response from a remotely located server. Otherwise, the device is automatically locked.

[0009] In different embodiments, the device can actively request a heartbeat response by sending an initial heartbeat request message to the server, or the device can passively receive non-prompted heartbeat responses from the server.

Either way, the received heartbeat response can permit the device to operate for an additional interval. Shifting the device from a locked state back to an unlocked state can require the receipt of an unlock command from a remotely located server. Accordingly, the device is unable to be utilized for any significant duration unless it is able to periodically receive heartbeat responses from one or more remotely located servers.

[0010] The present invention can be implemented in accordance with numerous aspects consistent with material presented herein. For example, one aspect of the present invention can include a method for automatically locking a client. The method can include a step of a client automatically establishing a heartbeat interval. A determination can be automatically made regarding whether a proper server response is received within the heartbeat interval. When no proper response is received, the client can be automatically placed in a locked state. All client functions accessible by a user other than those functions relating to unlocking the client can be disabled while the client is in the locked state. A remotely located server can unlock the client by conveying an unlock message to the client.

[0011] Another aspect of the present invention can include a method of restricting access to a computing device. The method can automatically generate a heartbeat event within a client. A determination can be made as to whether a server response is received by the client for the heartbeat event. The lock state of the client can be automatically altered based upon the determining step. In the method, a server response to the heartbeat event can be required to prevent the client from automatically entering a locked state.

[0012] Still another aspect of the present invention can include a storage space upon a machine-readable medium local to a client. The machine-readable medium can include code instructions for causing a machine to identify a heartbeat interval. A heartbeat timer can be started within the client. When a heartbeat response is received from a remotely located server, the heartbeat timer can be reset. When the heartbeat timer exceeds the heartbeat interval, the client can be automatically adjusted from an unlocked state to a locked state. All client functions accessible by a user other than those functions relating to unlocking the client can be disabled while the client is in the locked state.

[0013] It should be noted that various aspects of the invention can be implemented as a program for controlling computing equipment to implement the functions described herein, or a program for enabling computing equipment to perform processes corresponding to the steps disclosed herein. This program may be provided by storing the program in a magnetic disk, an optical disk, a semiconductor memory, or any other recording medium. The program can also be provided as a digitally encoded signal conveyed via a carrier wave. The described program can be a single program or can be implemented as multiple subprograms, each of which interact within a single computing device or interact in a distributed fashion across a network space.

[0014] It should also be noted that the methods detailed herein can also be methods performed at least in part by a service agent and/or a machine manipulated by a service agent in response to a service request.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] There are shown in the drawings, embodiments which are presently preferred, it being understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown.

[0016] FIG. 1 is a schematic diagram of a system for restricting devices using a heartbeat in accordance with an embodiment of the inventive arrangements disclosed herein.

[0017] FIG. 2 is a flow chart of a method for restricting devices using a heartbeat in accordance with an embodiment of the inventive arrangements disclosed herein.

[0018] FIG. 3 is a flow chart of a method in which a service agent can configure a system to implement a heartbeat that restricts client devices in accordance with an embodiment of the inventive arrangements disclosed herein.

DETAILED DESCRIPTION OF THE INVENTION

[0019] FIG. 1 is a schematic diagram of a system 100 for restricting devices using a heartbeat in accordance with an embodiment of the inventive arrangements disclosed herein. System 100 can include a client 110 and a client 111, each of which requires a periodic heartbeat response 116 from server 130 to prevent the client 110-111 from automatically entering a locked state. When in a locked state, the client 110-111 is unable to be utilized as intended by user 120 for any purpose other than attempting to unlock the client 110-111.

[0020] In one embodiment, data contained within client 110-111 can be secured when the client 110-111 enters a locked state. For example, data can be automatically deleted or shredded when the client 110-111 is locked. In another example, all data within the client 110-111 can be automatically encrypted when the client 110-111 enters a locked state. The data can be automatically decrypted, when the client 110-111 is placed in an unlocked state.

[0021] If data within client 110-111 is particularly sensitive, software can be installed that establishes an encrypted drive, where by default data within the drive is encrypted. When active or unlocked, a decryption key, stored in non-persistent memory such as RAM, can be used to dynamically decrypt data contained within the encrypted drive. Accordingly, accessing unencrypted data requires an affirmative step, which can only be performed when the client 110-111 is unlocked.

[0022] The client 110-111 can be any computing device upon which a heartbeat application 112 can be installed. The client 110-111 can include, but is not limited to, a computer, a personal data assistant (PDA), a mobile telephone, a laptop computer, a bar-code scanner, a media player, a wearable computing device, and other such computing devices. The client 110-111 can be configured so that user 120 is unable to remove the heartbeat application 112 from the client 110-111. The user 120 is also unable to prevent the heartbeat application 112 from entering a locked state in the absence of periodically received heartbeat responses 115 from server 130.

[0023] The heartbeat application 112 can establish a heartbeat interval and can include a heartbeat timer. Whenever the heartbeat timer exceeds the heartbeat interval, the client

110-111 can enter the locked state. The heartbeat response 116 can be used to reset the heartbeat timer. In one embodiment, the client 110-111 can actively solicit the server 130 for a heartbeat response 116 by conveying one or more heartbeat requests 114. In another embodiment, server 130 can broadcast or automatically convey heartbeat responses 116 to client 110-111 without an explicit heartbeat request 114 being made.

[0024] The heartbeat application 112 can be implemented within hardware, firmware, and/or software of the client 110-111. The heartbeat application 112 can be a daemon or background application executing on client 110 to which user 120 is not granted write, modify, or delete privileges. Heartbeat application 112 can also be a firmware or hardware based security process that can disable a critical portion of the client 110-111 when locked. For example, the heartbeat application 112 can disable all input/output ports other than a communication port to the server, when locked.

[0025] In one embodiment, the heartbeat application 112 can include a custom restriction profile. The profile can include one or more parameters that are able to be customized by an authorized individual. For example, a system administrator can change a heartbeat interval using the custom restriction profile. In another example, user 120 can modify the custom restriction profile to change the frequency with which heartbeat requests 114 are generated.

[0026] The heartbeat response 116 can include any type of message capable of resetting the heartbeat timer. It is common for the heartbeat response 116 to be implemented as a secure key or an encrypted pass code that is difficult for unauthorized users 120 to duplicate or ascertain. For example, the heartbeat response 116 can be implemented as a digital certificate. The heartbeat response 116 can also be implemented as one part of a public-private key combination, where a complimentary part is known by client 110-111. Conventional security practices and technologies can be utilized in conjunction with the heartbeat concept disclosed herein to ensure the heartbeat application 112 and automatic locking functions of the client 110-111 are not easily circumvented.

[0027] Server 130 can be any computing device capable of transmitting a heartbeat response 116 to the client 110-111. For example, server 130 can be a computer that receives heartbeat requests 114 from the client 110-111. Each heartbeat request 114 can include authorizing information, such as user 120 identification and password. The server 130 can determine whether user 120 is authorized to utilize client 110-111. If the use of client 110-111 by user 120 is authorized, the server 130 can convey a heartbeat response 116 to the client 110-111. For security reasons, system 100 can be configured so that heartbeat responses 116 expire, meaning that new and different heartbeat responses 116 are necessary after a designated time.

[0028] Once a client 110-111 has been locked, server 130 can generate an unlock command 118, which alters the lock state of the client 110-111. The unlock command 118 can be either generated responsive to an unlock request 117 or can be automatically generated by the server 130. While the unlock command 118 can be different from the heartbeat response 116, embodiments are contemplated where a single message from server 130 can function as both heartbeat response 116 and unlock command 118.

[0029] Server 130 can be communicatively linked to client 110-111 in any fashion that permits the exchange of digitally encoded information between the server 130 and the client 110-111. For example, the client 110-111 can be linked to server 130 through a network, which can be line-based or wireless. Information can be exchanged using any known communication protocol, such as Transmission Control Protocol/Internet Protocol (TCP/IP) based protocols, Universal Serial Bus (USB) protocols, BLUETOOTH protocols, Universal Plug and Play (UPnP) protocols, and the like.

[0030] In a common embodiment, server 130 and client 110-111 will communicate via a wireless communication system that has a limited range, denoted by wireless range 140. Range 140 can be centered upon one or more wireless transceivers. For example, when server 130 is wirelessly linked to client 110-111 through an 802.11 based protocol, the server can function as a wireless access point. In another example, multiple wireless transceivers can be established and combined to form any desired wireless range 140.

[0031] When outside the wireless range 140, client 110-111 can be unable to automatically communicate with server 130 and will therefore be unable to receive a heartbeat response 116 from the server 130. Consequently, the client 110-111 will enter a locked state. When a locked client 110-111 reenters the wireless range 140, the client 110-111 can receive the unlock command 118 from server 130. Thus, geographic boundaries in which clients 110-111 can be used are able to be established based upon a wireless communication range 140.

[0032] In one embodiment, system 100 can be implemented using a server 130 with robust authorization and transmission capabilities or using a server 130 with extremely limited computing resources. For example, server 130 can be implemented as a broadcasting beacon that intermittently broadcasts a key. The key can function as both heartbeat response 116 and unlock command 118. When clients 110-111 are outside the broadcast range of the beacon, no heartbeat response 116 is being received, which can cause the clients 110-111 to be placed in a locked state.

[0033] FIG. 2 is a flow chart of a method 200 for restricting devices using a heartbeat in accordance with an embodiment of the inventive arrangements disclosed herein. In one embodiment, the method 200 can be performed in the context of system 100.

[0034] Method 200 can begin in step 205, where a client is activated. Activation of a client can occur when the client is powered on. In step 210, a heartbeat application can be executed upon the client. In one arrangement, the instantiation of the heartbeat application can occur in a non-preemptible fashion, such as occurring as a Power On Self Test (POST) step of the client. In step 215, the heartbeat application can establish a heartbeat interval. In step 220, a heartbeat timer can be initialized.

[0035] In step 225, a check can be performed to see if the client has received a heartbeat response from a server. If so, the method can proceed to step 230 where the response can be validated. If the response is validated, the method can loop to step 220, where the heartbeat timer can be reset. If no heartbeat response is received or if a received heartbeat response is not valid, the method can proceed to step 235.

[0036] In step 235, an optional expected response time can be implemented. The expected response time can be a time

limit less than the heartbeat interval that causes a heartbeat request to be issued from the client to a server. The server can be configured to respond to heartbeat requests with heartbeat responses when the heartbeat requests are issued by a valid user and when the client is communicatively linked to (or within a communication range of) the server.

[0037] In step 240, another check can be performed for the heartbeat response. When a response is received, the response can be validated, as shown in step 245. A valid response causes the method to loop to step 220, where the heartbeat timer is reset. Otherwise, the method proceeds to step 250.

[0038] In step 250, an optional retransmission time can be implemented. The retransmission time can result in another heartbeat request being conveyed to the server. The retransmission time can be continuously decreased for each retransmission iteration, as shown by step 255. Thus, clients can more frequently issue heartbeat requests as the heartbeat timer approaches the heartbeat interval.

[0039] In step 260, if the heartbeat interval is exceeded, the method can branch to step 280, where the client is placed in a locked state. If the heartbeat interval is not exceeded, the method can progress from step 260 to step 265. In step 265, a check for a heartbeat response can be performed. A received response can be validated in step 270. If a valid heartbeat response is received, the method can loop from step 270 to step 220, where the heartbeat timer is reset. If no valid heartbeat response is received, the method can progress to step 275, where the heartbeat request can be retransmitted. The method can loop from step 275 to step 255.

[0040] Once the client has been placed in a locked state (step 280), the client can remain in that locked state until a valid unlock command is received (step 285). In step 290, the unlock command can place a client in an unlocked state. Upon entering the unlocked state, a new heartbeat timer can be initialized for the client. Hence, the method can loop from step 290 to step 220.

[0041] FIG. 3 is a flow chart of a method 300 in which a service agent can configure a system to implement a heartbeat that restricts client devices in accordance with an embodiment of the inventive arrangements disclosed herein. Method 300 can be performed in the context of system 100.

[0042] Method 300 can begin in step 305, when a customer initiates a service request. The service request can be a request for a service agent to configure a new system, such as system 100, for the client. The service request can also be a request to troubleshoot a problem with a client access system. For example, the service request can be a request to unlock a currently locked client, which is not responding to an unlock command issued by a heartbeat server.

[0043] In step 310, a human agent can be selected to respond to the service request. In step 315, the human agent can analyze a customer's current system and can develop a solution. The solution can include the acquisition and deployment of additional hardware, such as deployment of one or more heartbeat servers and/or wireless access points for wireless communication with a heartbeat server.

[0044] In step 320, the human agent can use one or more computing devices to perform or to cause the computer

device to perform the steps of method 200. For example, the agent can utilize agent specific software/hardware that functions as a skeleton or master key to unlock a locked device (steps 285, 290).

[0045] In optional step 325, the human agent can configure the customer's computer in a manner that the customer or clients of the customer can perform one or more steps of method 200 in the future. For example, the service agent can load and configure a heartbeat server and can deploy heartbeat applications upon customer owned client machines so that the clients and server automatically perform steps 210-290. In step 330, the human agent can complete the service activities.

[0046] It should be noted that while the human agent may physically travel to a location local to adjust the customer's computer or application server, physical travel may be unnecessary. For example, the human agent can use a remote agent to remotely manipulate the customer's heartbeat server or a customer owned client.

[0047] The present invention may be realized in hardware, software, or a combination of hardware and software. The present invention may be realized in a centralized fashion in one computer system or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system or other apparatus adapted for carrying out the methods described herein is suited. A typical combination of hardware and software may be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein.

[0048] The present invention also may be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which when loaded in a computer system is able to carry out these methods. Computer program in the present context means any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following: a) conversion to another language, code or notation; b) reproduction in a different material form.

[0049] This invention may be embodied in other forms without departing from the spirit or essential attributes thereof. Accordingly, reference should be made to the following claims, rather than to the foregoing specification, as indicating the scope of the invention.

What is claimed is:

1. A method of automatically locking a client comprising:
 - a client automatically establishing a heartbeat interval;
 - automatically determining whether a proper server response is received within the heartbeat interval; and
 - when no proper response is received, automatically placing the client in a locked state, wherein all client functions accessible by a user other than those functions relating to unlocking the client are disabled while the client is in the locked state, and wherein unlocking the client requires a remotely located server to provide an unlock message to the client.

2. The method of claim 1, wherein the placing step further comprises:

- automatically securing data contained within the client so that the secured data is not accessible while the client is in a locked state.

3. The method of claim 1, wherein the client and the remotely located server both include a wireless communication ability, wherein messages including the server response and the unlock message are wirelessly exchanged between the client and the remotely located server.

4. The method of claim 1, wherein a communication range is established within which the client is able to become communicatively linked to a server configured to provide heartbeat responses to at least one client to prevent the at least one client from entering a locked state, wherein the client is unable to receive the proper server response when located outside the communication range.

5. The method of claim 4, wherein the communication range is based upon a range of a wireless communication network to which the server is communicatively linked.

6. The method of claim 1, wherein said steps of claim 1 are performed by at least one machine in accordance with at least one computer program having a plurality of code sections that are executable by the at least one machine.

7. The method of claim 1, wherein the steps of claim 1 are performed by at least one of a service agent and a computing device manipulated by the service agent, the steps being performed in response to a service request.

8. A method of restricting access to a computing device comprising:

- automatically generating a heartbeat event within a client;
- determining whether a server response is received by the client for the heartbeat event; and

- automatically altering a lock state of the client based upon the determining step, wherein a server response to the heartbeat event is required to prevent the client from automatically adjusting from an unlocked state to a locked state.

9. The method of claim 8, further comprising:

- establishing a custom restriction profile upon the client, wherein the determining step is based upon the restriction profile.

10. The method of claim 9, further comprising:

- authenticating a user for the client; and

- ascertaining that the user possesses privileges to modify the custom restriction profile, wherein the client includes an interface through which the authenticated user is able to configure the custom restriction profile.

11. The method of claim 8, wherein the altering step alters the lock state of the client to a locked state, and wherein the client is configured to remain in the locked state until a communication pathway is established between the client and the server and until the server provides an unlock response to the client via the communication pathway.

12. The method of claim 11, wherein the client iteratively polls the server to receive the unlock response.

13. The method of claim 11, wherein all client functions accessible by a user other than those functions relating to unlocking the client are disabled while the client is in the locked state.

14. The method of claim 8, further comprising:
 responsive to the heartbeat event, the client automatically attempting to wirelessly transmit a heartbeat message to which the server response is expected, wherein the server response prevents the client from automatically adjusting from the unlocked state to the locked state.

15. The method of claim 14, further comprising:
 identifying an expected response time and a retransmission time, wherein the retransmission time is less than the expected response time;

when the client fails to receive the server response to the heartbeat message within the expected response time, the client retransmitting the heartbeat message; and

when the client fails to receive the server response to the retransmitted heartbeat message within the retransmission time, the client executing at least one of the altering step and a step of again retransmitting the heartbeat message.

16. The method of claim 8, wherein said steps of claim 8 are performed by at least one machine in accordance with at least one computer program having a plurality of code sections that are executable by the at least one machine.

17. The method of claim 8, wherein the steps of claim 8 are performed by at least one of a service agent and a computing device manipulated by the service agent, the steps being performed in response to a service request.

18. A storage space upon a machine-readable medium local to a client, the machine-readable medium comprising a plurality of code instructions for causing a machine to perform the steps of:

identifying a heartbeat interval;

starting a heartbeat timer within the client;

when a heartbeat response is received from a remotely located server, resetting the heartbeat timer; and

when the heartbeat timer exceeds the heartbeat interval, automatically adjusting the client from an unlocked state to a locked state, wherein all client functions accessible by a user other than those functions relating to unlocking the client are disabled while the client is in the locked state.

19. The storage space of claim 18, wherein the client is configured so that a user of the client is unable to disable the heartbeat timer and is unable to prevent the client from entering the locked state in absence of a heartbeat response being received from the remotely located server.

20. The storage space of claim 18, wherein the identifying, starting, and adjusting steps are performed as a background process executing upon the client, wherein users of the device are not authorized to remove the background process and are not authorized to disable the background process.

* * * * *

APPENDIX B-9



(19) **United States**

(12) **Patent Application Publication**
Natarajan et al.

(10) **Pub. No.: US 2007/0294380 A1**

(43) **Pub. Date: Dec. 20, 2007**

(54) **SYSTEM AND METHOD FOR PERIODIC
SERVER-TO-CLIENT DATA DELIVERY**

Publication Classification

(51) **Int. Cl.**
G06F 15/173 (2006.01)
(52) **U.S. Cl.** **709/223**
(57) **ABSTRACT**

(75) Inventors: **Giri Natarajan**, La Palma, CA (US); **Silvy Wilson**, Rancho Santa Margar., CA (US); **William Su**, Riverside, CA (US)

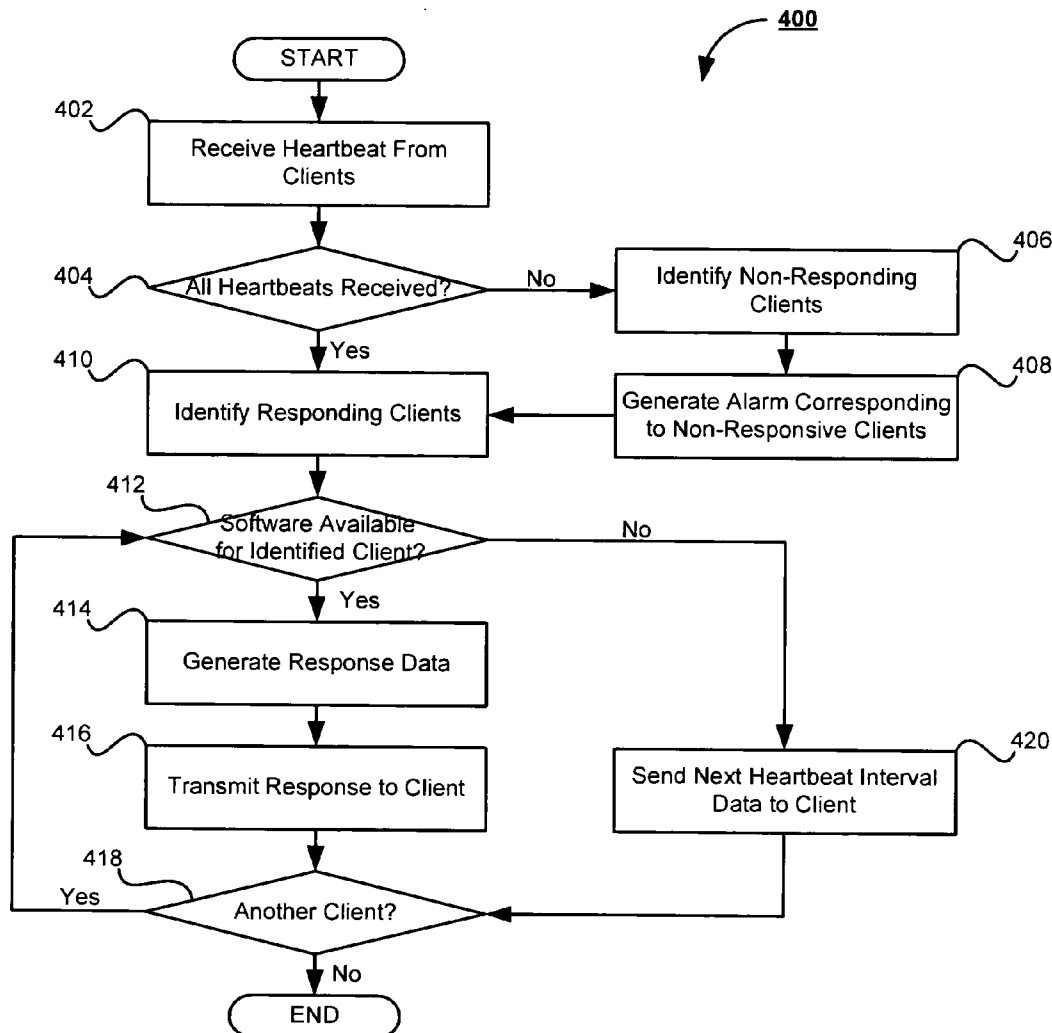
A server-to-client data delivery system and method is provided. The system includes a server capable of providing a variety of services to one or more client devices. Each client device periodically transmits a heartbeat signal to the server over an associated network including client device identification data. When the server fails to receive a heartbeat from an associated client device, an alert signal is generated indicating the non-responsiveness of the associated client device. For each responsive client device, the server performs an identification of the device and determines whether new or updated software, or supplemental data, is available for the client. The server then generates response data including the next heartbeat interval and the software or supplemental data. The response data is then transmitted to the responsive and identified client, whereupon the server proceeds to perform the same process for each responsive and identified client.

Correspondence Address:
TUCKER ELLIS & WEST LLP
1150 HUNTINGTON BUILDING, 925 EUCLID AVENUE
CLEVELAND, OH 44115-1414

(73) Assignees: **Kabushiki Kaisha Toshiba;**
Toshiba Tec Kabushiki Kaisha

(21) Appl. No.: **11/452,778**

(22) Filed: **Jun. 14, 2006**



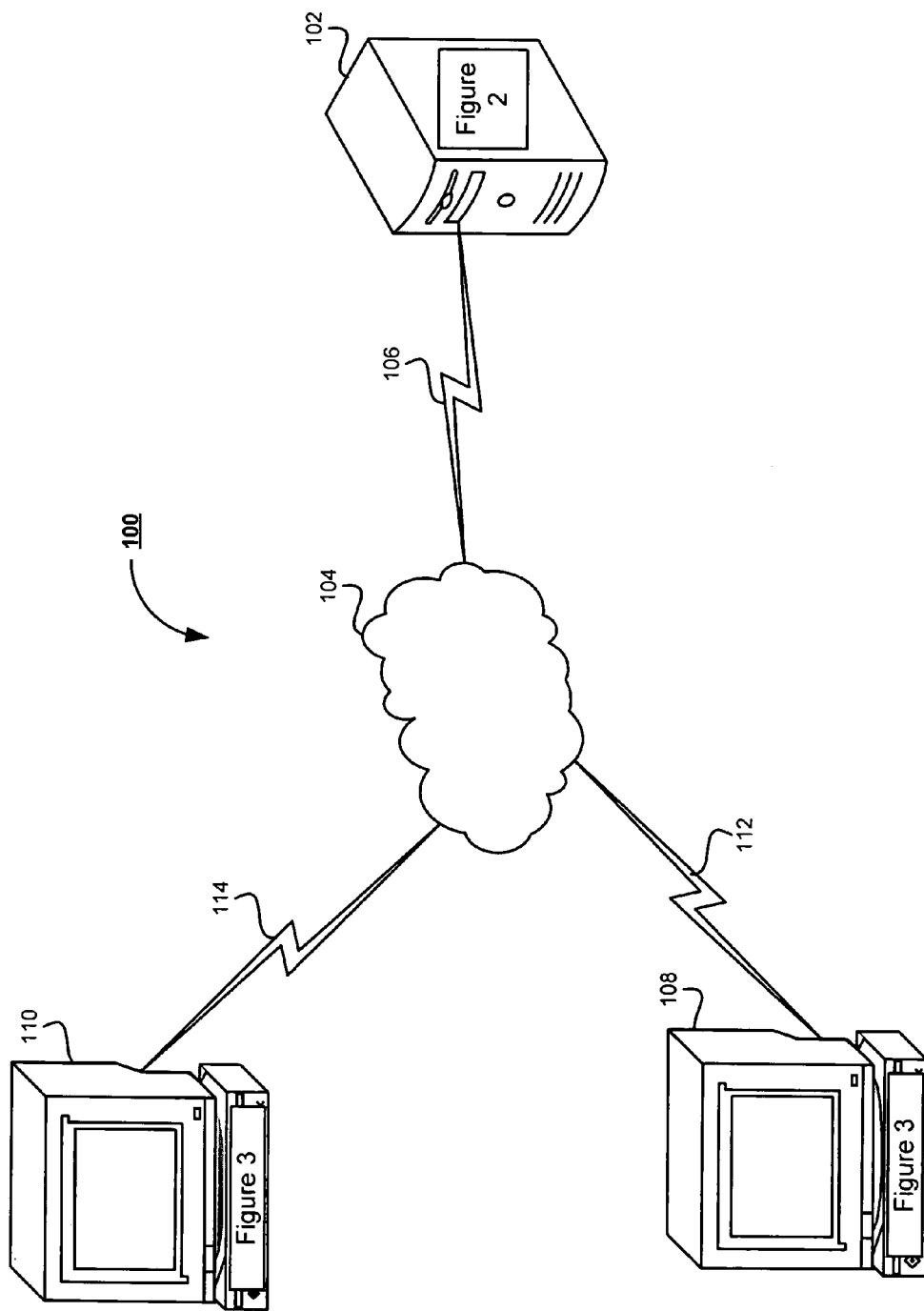


Figure 1

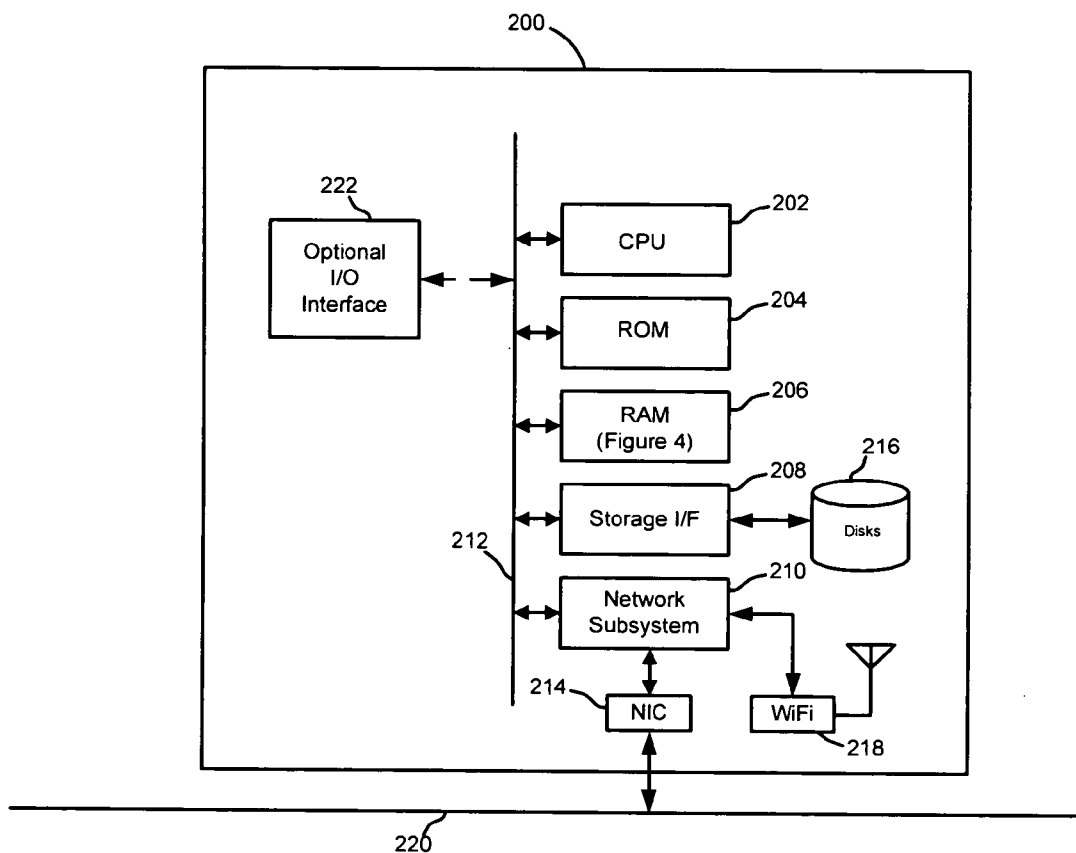


Figure 2

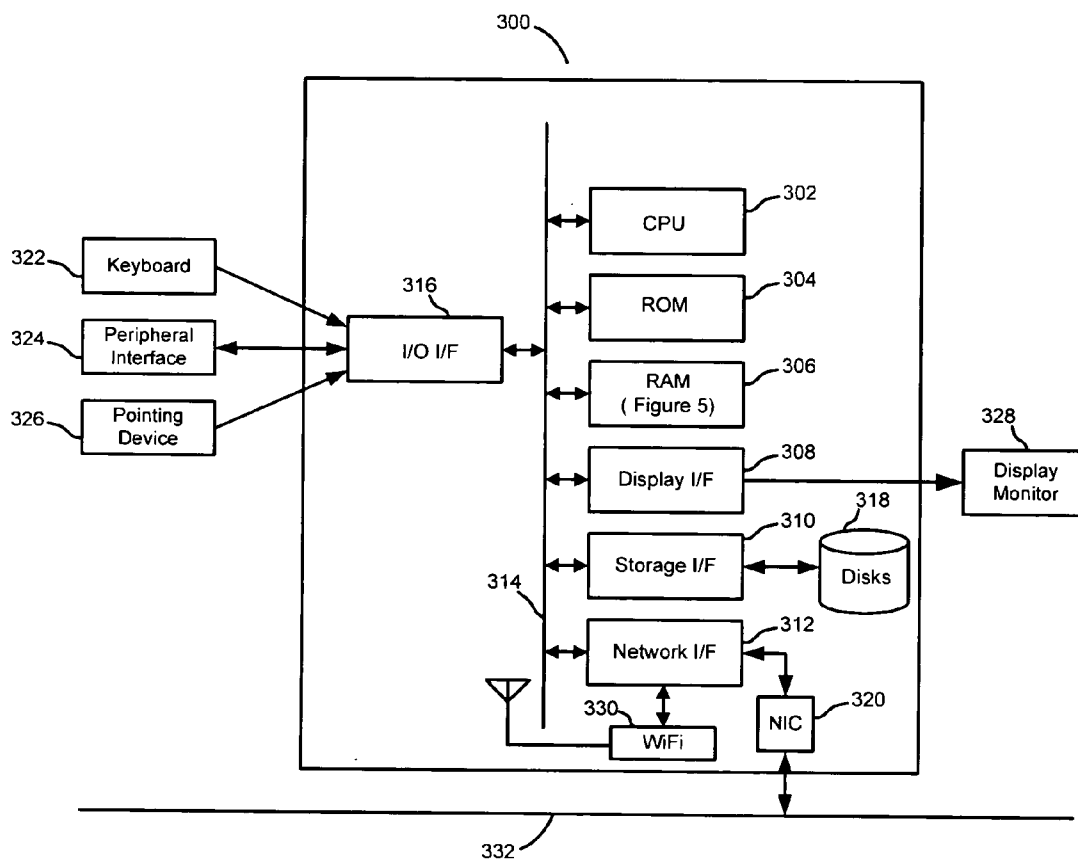


Figure 3

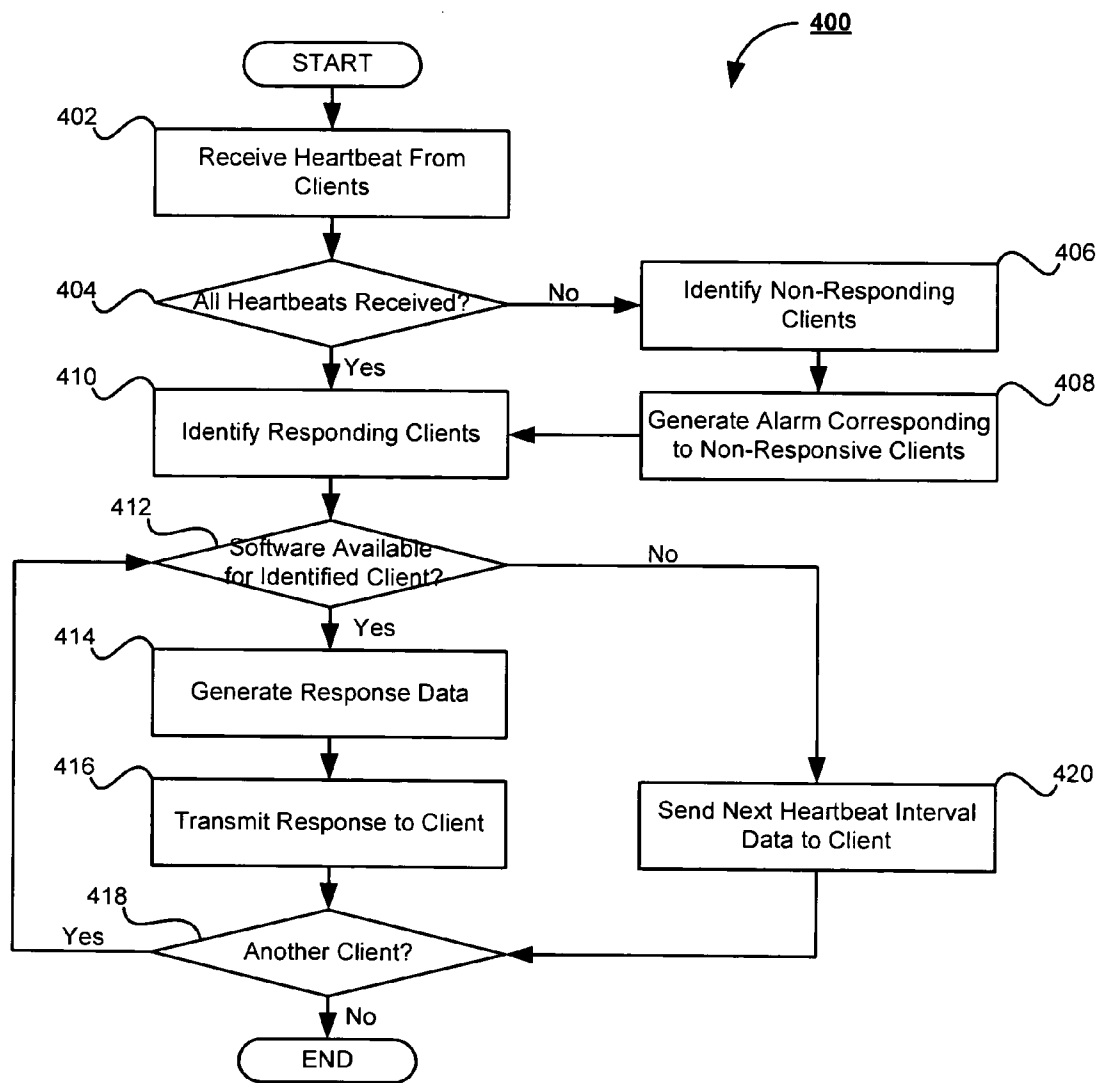


Figure 4

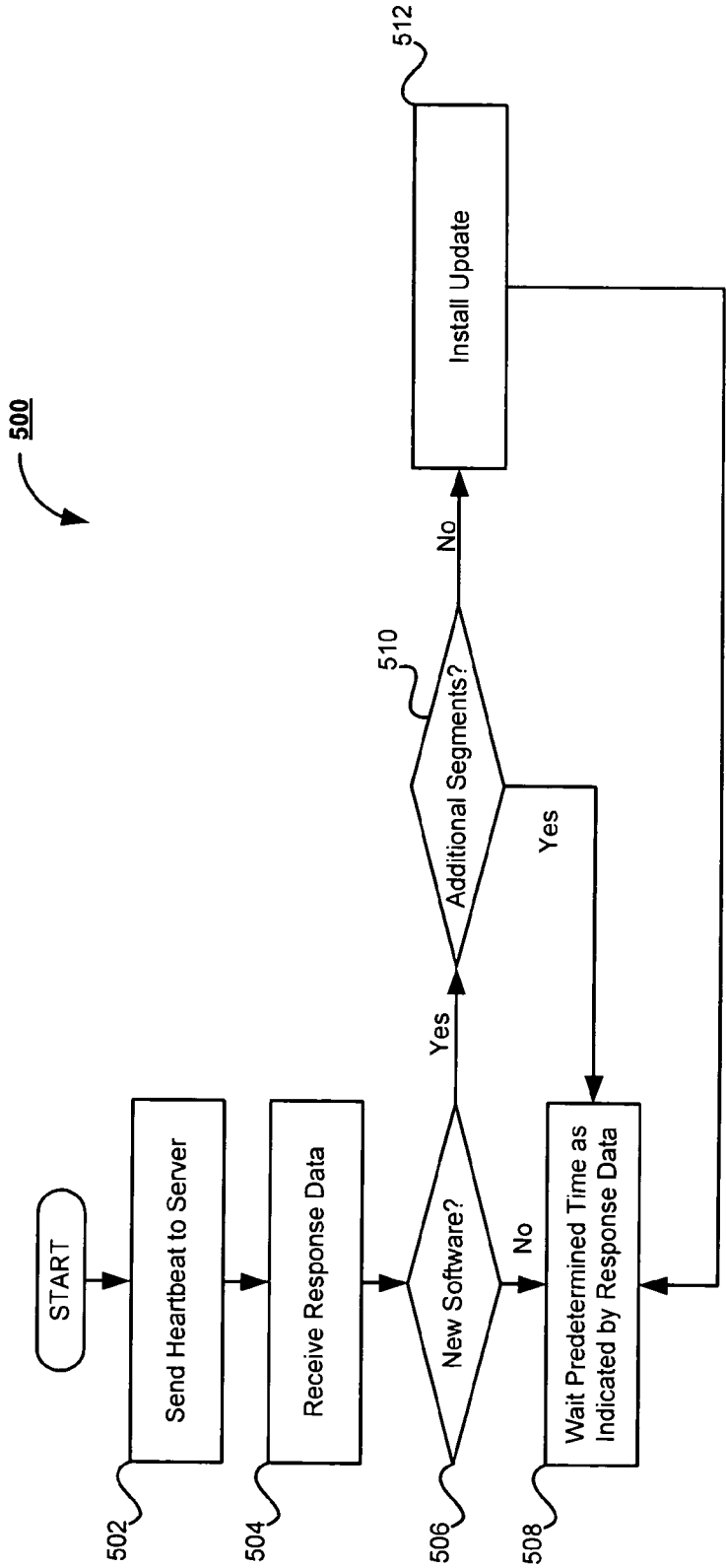


Figure 5

SYSTEM AND METHOD FOR PERIODIC SERVER-TO-CLIENT DATA DELIVERY

BACKGROUND OF THE INVENTION

[0001] The subject application is directed to a system and method for server-to-client data delivery. More particularly, the subject application is directed to a system wherein a client device periodically transmits information or heartbeat data to a server to inform the server that the client is presently accessing the server. In response to such heartbeat data transmission, the server will periodically transmit software, updates, data, or other supplemental information to such client.

[0002] In a typical web-based data communication between a client and a server, a server has no information as to presence or availability of a client until such time as the client initiates a request for services to that server. There are frequent requirements to update software, data, drivers, operating systems, and the like on networked workstations. While any workstation requires periodic updates, this is also a requirement in workstations that are used in connection with document processing operations.

[0003] In prior, network-based update procedures, it is incumbent on a workstation to inquire as to whether any new, modified or updated software, data, drivers, operating systems, and the like are available on a server. If a server has such information, it cannot determine a presence of a workstation or a need for transmission until it receives information from the workstation first. As such, there is a need for a system and method wherein a client periodically informs the server of its presence on the network.

[0004] The subject system seizes upon receipt of a periodic heartbeat pulse to selectively allow for communicating of such new or updated data or executable code to a workstation in need of the same.

SUMMARY OF THE INVENTION

[0005] In accordance with the subject application, there is provided a system and method for server-to-client data delivery.

[0006] Further, in accordance with the subject application, there is provided a system wherein a client device periodically transmits information or heartbeat data to a server to inform the server that the client is presently accessing the server, wherein in response to such heartbeat data transmission, the server will periodically transmit software, updates, data, or other supplemental information to such client.

[0007] Still further, in accordance with the subject application, there is provided a system and method seizes upon receipt of a periodic heartbeat pulse to selectively allow for communicating of such new or updated data or executable code to a workstation in need of the same.

[0008] Still further, in accordance with the subject application, there is provided a server-to-client data delivery system. The system comprises input means adapted for periodically receiving heartbeat data from each of a plurality of associated workstations via a network. The heartbeat data includes identification data representative of an identity of each of the associated workstations. The system also comprises testing means adapted for testing received request data to identify delivery data targeted for at least one of the associated workstation as well as alarm means adapted for generating an alarm signal corresponding to each worksta-

tion from which no heartbeat data has been received for a preselected time period. The system also includes means adapted for selectively generating response data responsive to received heartbeat data inclusive of identified delivery data targeted for the at least one associated workstation and means adapted for communicating response data to the at least one associated workstation in accordance with identification data associated therewith.

[0009] Still further, in accordance with the subject application, there is provided a server-to-client data delivery method. The method includes the steps of periodically receiving heartbeat data from each of a plurality of associated workstations via a network, wherein the heartbeat data including identification data representative of an identity of each of the associated workstations. The method also comprises the steps of testing received request data to identify delivery data targeted for at least one of the associated workstation and generating an alarm signal corresponding to each workstation from which no heartbeat data has been received for a preselected time period. If delivery data is detected, then the method selectively generates response data responsive to received heartbeat data inclusive of identified delivery data targeted for the at least one associated workstation and communicates response data to the at least one associated workstation in accordance with identification data associated therewith.

[0010] In a preferred embodiment, the associated workstation is comprised of a document processing kiosk and the delivery data includes executable code adapted for operation thereof. Preferably, the executable code performs an update of software located on document processing kiosk.

[0011] In a preferred embodiment, the system and method further include the ability to communicate a plurality of response data sets to the at least one associated workstation corresponding to a consecutive plurality of received heartbeat data corresponding thereto, such that the at least one workstation receives delivery data in a plurality of segments conjoined at the associated workstation.

[0012] Still other advantages, aspects and features of the subject application will become readily apparent to those skilled in the art from the following description wherein there is shown and described a preferred embodiment of the subject application, simply by way of illustration of one of the best modes best suited for to carry out the subject application. As it will be realized, the subject application is capable of other different embodiments and its several details are capable of modifications in various obvious aspects all without departing from the scope of the subject application. Accordingly, the drawing and descriptions will be regarded as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The subject application is described with reference to certain figures, including:

[0014] FIG. 1 which is an overall system diagram for server-to-client data delivery system according to the subject application;

[0015] FIG. 2 is a block diagram illustrating server hardware for use in the system for server-to-client data delivery according to the subject application;

[0016] FIG. 3 is a block diagram illustrating workstation hardware for use in the system for server-to-client data delivery according to the subject application;

[0017] FIG. 4 is a flowchart illustrating the method for a server-to-client data delivery from a server point of view according to the subject application; and

[0018] FIG. 5 is a flowchart illustrating the method for a server-to-client data delivery from a client point of view according to the subject application.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0019] The subject application is directed to a system and method for server-to-client data delivery. In particular, the subject application is directed to a system and method wherein a client device periodically transmits information or heartbeat data to a server to inform the server that the client is presently accessing the server, wherein in response to such heartbeat data transmission, the server will periodically transmit software, updates, data, or other supplemental information to such client. More particularly, the subject application is directed to a system and method that seizes upon receipt of a periodic heartbeat pulse to selectively allow for the communication of new or updated executable code to a workstation in need of the same.

[0020] Referring now to FIG. 1, there is shown a block diagram of a system 100 in accordance with the subject application. As shown in FIG. 1, the system 100 includes a server 102 in data communication with a distributed communications environment 104 via a suitable communications link 106. It will be appreciated by those skilled in the art that the server is representative of any computer hardware employed in a server-type role in a client-server relationship. In the preferred embodiment, the server 102 is suitably adapted to provide, or host, a variety of applications and services, which are used by one or more client devices. Preferably, the server 102 is capable of providing a variety of web-based services, including, for example and without limitation, remote access, remote storage, document processing operations, print job generation, electronic mail, document management services, and the like. The functioning of the server 102 will be better understood in conjunction with the block diagram illustrated in FIG. 2 and discussed in greater detail below.

[0021] In the preferred embodiment, the distributed communications environment 104 is a computer network, suitably adapted to enable the exchange of data between two or more electronic devices. In accordance with one aspect of the subject application, the network 104 is a distributed network, including, for example and without limitation, the Internet, wide area network, or the like. It will be appreciated by those skilled in the art that suitable networks include, for example and without limitation, a proprietary communications network, a local area network, a personal area network, an intranet, and the like.

[0022] Communications between the distributed network 104 and the server 102 are advantageously accomplished using the communications link 106. As will be appreciated by those skilled in the art, the communications link 106 is any data communication medium, known in the art, capable of enabling the exchange of data between two electronic devices. The communications link 106 is any suitable channel of data communications known in the art including, but not limited to wireless communications, for example and without limitation, Bluetooth, WiMax, 802.11a, 802.11b, 802.11g, 802.11(x), a proprietary communications network, infrared, optical, the public switched telephone network, or

any suitable wireless data transmission system, or wired communications known in the art.

[0023] The system 100 also includes a first client device and a second client device, illustrated in FIG. 1 as the first computer workstation 108 and the second computer workstation 110. It will be appreciated by those skilled in the art that the client devices, e.g., first workstation 108 and second workstation 110 are shown in FIG. 1 as computer workstations for example purposes. As the skilled artisan will understand, the workstations 108 and 110 shown in FIG. 1 are representative of any computing device known in the art, including, for example and without limitation, workstation, a document processing kiosk, a personal computer, a personal data assistant, a web-enabled cellular telephone, a smart phone, or other web-enabled electronic device suitably capable of generating and/or transmitting electronic document data to a multifunctional peripheral device. It will be understood by those skilled in the art that the client device, or workstation 108 or 110 is suitably capable of implementation as a controller associated with a document processing device (not shown). The skilled artisan will appreciate that such an embodiment is in accordance with the methodologies and systems described and claimed herein. The functioning of the first computer workstation 108 and the second computer workstation 110 will better be understood in conjunction with the block diagram illustrated in FIG. 3, discussed in greater detail below.

[0024] In the preferred embodiment, the workstations 108 and 110 are in data communication with the network 104 via suitable communications links 112 and 114, respectively. As will be understood by those skilled in the art, the communications links 112 and 114 are any suitable communications channels known in the art including, for example and without limitation, WiMax, 802.11a, 802.11b, 802.11g, 802.11(x), Bluetooth, the public switched telephone network, a proprietary communications network, infrared, optical, or any other suitable wired or wireless data transmission communications known in the art. In the preferred embodiment of the subject application, the workstations 108 and 110 are advantageously equipped to facilitate the generation of service requests to be performed by the server 102.

[0025] Turning now to FIG. 2, illustrated is a representative architecture of a suitable server 200, shown in FIG. 1 as the server 102, on which operations of the subject system 100 are completed. Included is a processor 202, suitably comprised of a central processor unit. However, it will be appreciated that processor 202 may advantageously be composed of multiple processors working in concert with one another as will be appreciated by one of ordinary skill in the art. Also included is a non-volatile or read only memory 204 which is advantageously used for static or fixed data or instructions, such as BIOS functions, system functions, system configuration, and other routines or data used for operation of the server 200.

[0026] Also included in the server 200 is random access memory 206, suitably formed of dynamic random access memory, static random access memory, or any other suitable, addressable memory system. Random access memory provides a storage area for data instructions associated with applications and data handling accomplished by processor 202.

[0027] A storage interface 208 suitably provides a mechanism for volatile, bulk or long term storage of data associated with the server 200. The storage interface 208 suitably

uses bulk storage, such as any suitable addressable or serial storage, such as a disk, optical, tape drive and the like as shown as 216, as well as any suitable storage medium as will be appreciated by one of ordinary skill in the art.

[0028] A network interface subsystem 210 suitably routes input and output from an associated network allowing the server 200 to communicate to other devices. Network interface subsystem 210 suitably interfaces with one or more connections with external devices to the server 200. By way of example, illustrated is at least one network interface card 214 for data communication with fixed or wired networks, such as Ethernet, token ring, and the like, and a wireless interface 218, suitably adapted for wireless communication via means such as WiFi, WiMax, wireless modem, cellular network, or any suitable wireless communication system. It is to be appreciated however, that the network interface subsystem suitably utilizes any physical or non-physical data transfer layer or protocol layer as will be appreciated by one of ordinary skill in the art. In the illustration, the network interface 214 is interconnected for data interchange via a physical network 220, suitably comprised of a local area network, wide area network, or a combination thereof.

[0029] Data communication between the processor 202, read only memory 204, random access memory 206, storage interface 208 and network subsystem 210 is suitably accomplished via a bus data transfer mechanism, such as illustrated by bus 212.

[0030] Suitable executable instructions on the server 200 facilitate communication with a plurality of external devices, such as workstations, document processing devices, other servers, or the like. While, in operation, a typical server operates autonomously, it is to be appreciated that direct control by a local user is sometimes desirable, and is suitably accomplished via an optional input/output interface 222 as will be appreciated by one of ordinary skill in the art.

[0031] Referring now to FIG. 3, illustrated is a hardware diagram of a suitable workstation 300 for use in connection with the subject system 100. The skilled artisan will appreciate that the workstation 300 depicted in FIG. 3 is representative of both the first workstation 108 and the second workstation 110, shown in FIG. 1. A suitable workstation includes a processor unit 302 which is advantageously placed in data communication with read only memory 304, suitably non-volatile read only memory, volatile read only memory or a combination thereof, random access memory 306, display interface 308, storage interface 310, and network interface 312. In a preferred embodiment, interface to the foregoing modules is suitably accomplished via a bus 314.

[0032] Read only memory 304 suitably includes firmware, such as static data or fixed instructions, such as BIOS, system functions, configuration data, and other routines used for operation of the workstation 300 via CPU 302.

[0033] Random access memory 306 provides a storage area for data and instructions associated with applications and data handling accomplished by processor 302.

[0034] Display interface 308 receives data or instructions from other components on bus 314, which data is specific to generating a display to facilitate a user interface. Display interface 308 suitably provides output to a display terminal 326, suitably a video display device such as a monitor, LCD, plasma, or any other suitable visual output device as will be appreciated by one of ordinary skill in the art.

[0035] Storage interface 310 suitably provides a mechanism for non-volatile, bulk or long term storage of data or instructions in the workstation 300. Storage interface 310 suitably uses a storage mechanism, such as storage 318, suitably comprised of a disk, tape, CD, DVD, or other relatively higher capacity addressable or serial storage medium.

[0036] Network interface 312 suitably communicates to at least one other network interface, shown as network interface 320, such as a network interface card, and wireless network interface 330, such as a WiFi wireless network card. It will be appreciated that by one of ordinary skill in the art that a suitable network interface is comprised of both physical and protocol layers and is suitably any wired system, such as Ethernet, token ring, or any other wide area or local area network communication system, or wireless system, such as WiFi, WiMax, or any other suitable wireless network system, as will be appreciated by one of ordinary skill in the art. In the illustration, the network interface 320 is interconnected for data interchange via a physical network 332, suitably comprised of a local area network, wide area network, or a combination thereof.

[0037] An input/output interface 316 in data communication with bus 314 is suitably connected with an input device 322, such as a keyboard or the like. Input/output interface 316 also suitably provides data output to a peripheral interface 324, such as a USB, universal serial bus output, SCSI, Firewire (IEEE 1394) output, or any other interface as may be appropriate for a selected application. Finally, input/output interface 316 is suitably in data communication with a pointing device interface 328 for connection with devices, such as a mouse, light pen, touch screen, or the like. The skilled artisan will appreciate that the use of the workstation 300 herein is for example purposes only. It will be apparent to those skilled in the art that the subject application is capable of incorporating the components described above and the methodologies described below on any myriad of computing devices, including, for example and without limitation, a controller associated with a document processing device, a laptop computer, a personal computer, a personal data assistant, a web-enabled cellular telephone, a proprietary portable electronic communication device, or the like.

[0038] In operation, the workstations 108 and 110 routinely, and upon the expiration of a predetermined time period, transmit a heartbeat signal to the server 102 indicating that the workstations 108 and 110 are accessing services provided by the server 102. Preferably, the heartbeat signal sent by the workstations 108 and 110 include data representative of the identity of the sending device, i.e., workstation 108 or workstation 110. When a predetermined period of time has expired, as set by a response from the server 102 to the receipt of the preceding heartbeat signal, the workstation 108 sends a heartbeat signal to the server 102 via the communications network 104. Similarly, the workstation 110 sends a heartbeat signal to the server 102 upon the expiration of a predetermined period of time, as set in the response to the preceding heartbeat signal.

[0039] The server 102 routinely receives heartbeat signals from multiple workstations associated with the services provided by the server 102. Preferably, the signals received from each workstation 108 or 110 serve to identify the device from among all devices sending such signals to the server 102. In the event that the server 102 does not receive

a heartbeat signal from one of the devices, e.g., workstation **110**, the server **102** issues an alert signal indicating the non-responsiveness of the device **110** to an administrator, service log, technical support, or the like.

[0040] For each heartbeat signal received from a workstation, the server **102** identifies the originating device, e.g., responsive workstation **108**. Based upon this identification, the server **102** determines whether executable code, a software update, upgrade, or other supplemental data, i.e., delivery data, is available for the workstation **108**. When no such software is available, the server **102** generates and transmits data representing the next heartbeat transmission interval to the workstation **108**. The server **102** then determines if additional clients remain for determination of the availability of updates. When an update is available, the server **102** generates response data including the new or updated software, as well as the next interval for heartbeat transmission, to the workstation **108**. When the software is exceptionally large, it is segmented by the server **102** and transmitted in successive responses to heartbeat signals until all such segments have been received by the workstation **108**. Following receipt of the entire software update, the workstation **108** installs the new or upgraded software. The workstation **108** then transmits a heartbeat signal upon the expiration of the predetermined period of time, as set by the last response from the server **102**. In one particular embodiment, the software is transmitted as executable code, the execution of which installs the updates, new software, upgrades, supplemental data, or the like.

[0041] The functioning of the system **100** and the components described above in accordance with FIG. 1, FIG. 2, and FIG. 3 will better be understood in conjunction with the method illustrated in FIG. 4 and FIG. 5. Referring now to FIG. 4, there is shown a server-side view of the server-to-client data delivery method according to the subject application. As shown in FIG. 4, the flowchart **400** illustrates the server-side of operations of the method in accordance with the subject application. Beginning at step **402**, the server **102** receives a heartbeat signal from each of a plurality of workstations associated with the services provided by the server **102**. As previously discussed, the services provided by the server **102** include, but are not limited to, web-based services, including, for example and without limitation, remote access, security verification services, quota management, remote storage, document processing operations, print job generation, electronic mail, document management services, and the like. In accordance with the preferred embodiment of the subject application, each workstation **108** and **110** are instructed to periodically send a heartbeat signal to the server **102** indicating that the device is accessing the server **102**. The period of transmission is predetermined during initial connection of the device to the server **102**, set by a response to the heartbeat signal from the server **102**, or any other manner known in the art.

[0042] The server **102** then determines, at step **404**, whether heartbeat signals have been received from all workstations **108** and **110** associated with the server **102**. When the server **102** determines that one or more workstations are non-responsive, i.e., failed to transmit a heartbeat signal within the preselected period of time, the server **102** identifies the non-responsive clients at step **406**. Following identification, the server **102** generates an alert signal corresponding to each non-responsive device. Flow then proceeds to step **410**, whereupon the responsive clients are

identified based upon the received heartbeat signals. It will be appreciated by those skilled in the art that a positive determination that all clients have responded at step **404** prompts flow to proceed to the identification of each client at step **410**.

[0043] After the server **102** has identified each responsive client, operations continue to step **412**, whereupon a determination is made whether an update is available for a workstation. Those skilled in the art will appreciate that the update, new software, or supplemental data determination is made for each responsive and identified workstation. When no such update or new software is available for the identified workstation, flow proceeds to step **420**, whereupon the next heartbeat transmission time is set and sent to the workstation. A determination is then made at step **418** whether another workstation remains for upgrade determination. When no additional workstations remain, operations terminate as the server **102** awaits the receipt of the next batch of heartbeat signals from the associated workstations to begin the process again at step **402**. When additional workstations remain, flow returns to step **412**, whereupon the next responsive and identified workstation is analyzed to determine whether an update or upgrade is available for the workstation.

[0044] When it is determined, at step **412**, that new software is available for the responsive and identified workstation, flow proceeds to step **414**, whereupon response data is generated by the server **102** inclusive of the next heartbeat time period, and the update, upgraded software, or executable code for a first workstation. The response data is then transmitted to the first workstation at step **416**. It will become apparent to those skilled in the art that in the event that the software is too large for a single "piggy-back" transmission, i.e., attachment to the heartbeat interval setting transmission, the server **102** is capable of segmenting the software into smaller components, which are then transmitted individually or as groups, at subsequent heartbeat intervals to the corresponding workstation, thereby allowing the client to receive the entire update and install following receipt of the last segment. Returning to step **416**, following transmission of the response data to the workstation, flow proceeds to step **418**, whereupon a determination is made whether another workstation remains for analysis during the current heartbeat time interval. When no additional workstations remain, the operation terminates until the receipt of the next batch of heartbeat signals at step **402**. When one or more additional workstations remain, operation returns to the determination step **412** for the next responsive and identified workstation.

[0045] Turning now to FIG. 5, there is shown a flowchart **500** illustrating the client side operation in accordance with the subject application. As shown in FIG. 5, the flowchart **300** illustrates the method of generating a heartbeat signal by a workstation and receiving software updates, upgrades, or supplemental data in accordance with the subject application. For purposes of explanation only, reference with respect to FIG. 5 will be explained using the workstation **108**. The skilled artisan will appreciate that the method depicted in FIG. 5 is applicable to each device associated with the services provided by the server **102**. Beginning at step **502**, the workstation **108** generates and transmits a heartbeat signal to the server **102**, indicating the identity of the workstation **108**, and that the workstation **108** is accessing the server **102** provided services. At step **504**, the

workstation **104** receives response data. In accordance with the preferred embodiment of the subject application, the response data includes a preselected time period, the expiration of which prompts the generation and transmission of a next heartbeat signal from the workstation **108** to the server **102**. In addition, the response data is advantageously capable of including new, updated, or upgraded software, supplemental data, executable code, or other delivery data.

[0046] The workstation **108** then determines, at step **506**, whether the response data includes new software or supplemental data. When no such software or data is indicated, flow proceeds to step **508**, whereupon the workstation **108** waits the preselected period of time, as set by the response data or preset by an administrator, before returning to step **502** and transmitting the next heartbeat signal. When software updates or supplemental data is included in the response data, as determined in step **506**, flow proceeds to step **510**, whereupon a determination is made whether additional data segments are required for installation of the new software or the supplemental data, or execution of the executable code. When additional segments are required, flow proceeds to step **508**, whereupon the workstation **108** waits the preselected period of time before transmitting the next heartbeat signal at step **502**. When all segments have been received, or when no additional segments are indicated, flow proceeds to step **512**, whereupon the new software is installed on the workstation **108**. The workstation **108** then progresses to step **508**, whereupon the device **108** waits the preselected period of time before transmitting the next heartbeat signal. The skilled artisan will appreciate that the method described in FIG. **3** is repeated for each responsive workstation associated with the server **102**.

[0047] The subject application extends to computer programs in the form of source code, object code, code intermediate sources and object code (such as in a partially compiled form), or in any other form suitable for use in the implementation of the subject application. Computer programs are suitably standalone applications, software components, scripts or plug-ins to other applications. Computer programs embedding the subject application are advantageously embodied on a carrier, being any entity or device capable of carrying the computer program: for example, a storage medium such as ROM or RAM, optical recording media such as CD-ROM or magnetic recording media such as floppy discs. The carrier is any transmissible carrier such as an electrical or optical signal conveyed by electrical or optical cable, or by radio or other means. Computer programs are suitably downloaded across the Internet from a server. Computer programs are also capable of being embedded in an integrated circuit. Any and all such embodiments containing code that will cause a computer to perform substantially the subject application principles as described, will fall within the scope of the subject application.

[0048] The foregoing description of a preferred embodiment of the subject application has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the subject application to the precise form disclosed. Obvious modifications or variations are possible in light of the above teachings. The embodiment was chosen and described to provide the best illustration of the principles of the subject application and its practical application to thereby enable one of ordinary skill in the art to use the subject application in various embodiments and with various modifications as are suited to the particular use

contemplated. All such modifications and variations are within the scope of the subject application as determined by the appended claims when interpreted in accordance with the breadth to which they are fairly, legally and equitably entitled.

What is claimed:

1. A server-to-client data delivery system comprising:
 - input means adapted for periodically receiving heartbeat data from each of a plurality of associated workstations via a network, the heartbeat data including identification data representative of an identity of each of the associated workstations;
 - testing means adapted for testing received identification data to identify delivery data targeted for at least one of the associated workstation;
 - alarm means adapted for generating an alarm signal corresponding to each workstation from which no heartbeat data has been received for a preselected time period;
 - means adapted for selectively generating response data responsive to received heartbeat data inclusive of identified delivery data targeted for the at least one associated workstation; and
 - means adapted for communicating response data to the at least one associated workstation in accordance with identification data associated therewith.
2. The server-to-client data delivery system of claim 1 wherein the associated workstation is comprised of a document processing kiosk.
3. The server-to-client data delivery system of claim 2 further comprising means adapted for communicating a plurality of response data sets to the at least one associated workstation corresponding to a consecutive plurality of received heartbeat data corresponding thereto, such that the at least one workstation receives delivery data in a plurality of segments conjoined at the associated workstation.
4. The server-to-client data delivery system of claim 1 wherein the delivery data includes executable code adapted for operation thereof.
5. The server-to-client data delivery system of claim 4 wherein the executable code performs an update of software located on document processing kiosk.
6. A server-to-client data delivery method comprising the steps of:
 - periodically receiving heartbeat data from each of a plurality of associated workstations via a network, the heartbeat data including identification data representative of an identity of each of the associated workstations;
 - testing received identification data to identify delivery data targeted for at least one of the associated workstation;
 - generating an alarm signal corresponding to each workstation from which no heartbeat data has been received for a preselected time period;
 - selectively generating response data responsive to received heartbeat data inclusive of identified delivery data targeted for the at least one associated workstation; and
 - communicating response data to the at least one associated workstation in accordance with identification data associated therewith.

7. The server-to-client data delivery method of claim 6 wherein the associated workstation is comprised of a document processing kiosk.

8. The server-to-client data delivery method of claim 7 further comprising the step of communicating a plurality of response data sets to the at least one associated workstation corresponding to a consecutive plurality of received heartbeat data corresponding thereto, such that the at least one workstation receives delivery data in a plurality of segments conjoined at the associated workstation.

9. The server-to-client data delivery method of claim 6 wherein the delivery data includes executable code adapted for operation thereof.

10. The server-to-client data delivery method of claim 9 wherein the executable code performs an update of software located on document processing kiosk.

11. A computer-implemented method for server-to-client data delivery comprising the steps of:

periodically receiving heartbeat data from each of a plurality of associated workstations via a network, the heartbeat data including identification data representative of an identity of each of the associated workstations;

testing received identification data to identify delivery data targeted for at least one of the associated workstation;

generating an alarm signal corresponding to each workstation from which no heartbeat data has been received for a preselected time period;

selectively generating response data responsive to received heartbeat data inclusive of identified delivery data targeted for the at least one associated workstation; and

communicating response data to the at least one associated workstation in accordance with identification data associated therewith.

12. The computer-implemented method for server-to-client data delivery of claim 11 wherein the associated workstation is comprised of a document processing kiosk.

13. The computer-implemented method for server-to-client data delivery of claim 12 further comprising the step of communicating a plurality of response data sets to the at least one associated workstation corresponding to a consecutive plurality of received heartbeat data corresponding thereto, such that the at least one workstation receives delivery data in a plurality of segments conjoined at the associated workstation.

14. The computer-implemented method for server-to-client data delivery of claim 11 wherein the delivery data includes executable code adapted for operation thereof.

15. The computer-implemented method for server-to-client data delivery of claim 14 wherein the executable code performs an update of software located on document processing kiosk.

* * * * *

APPENDIX B-10



US 20020165849A1

(19) **United States**

(12) **Patent Application Publication**
Singh et al.

(10) **Pub. No.: US 2002/0165849 A1**

(43) **Pub. Date: Nov. 7, 2002**

(54) **AUTOMATIC ADVERTISER NOTIFICATION FOR A SYSTEM FOR PROVIDING PLACE AND PRICE PROTECTION IN A SEARCH RESULT LIST GENERATED BY A COMPUTER NETWORK SEARCH ENGINE**

(76) **Inventors:** Narinder Pal Singh, Half Moon Bay, CA (US); Scott W. Snell, Hollywood, CA (US); Douglas T. Huffman, Altadena, CA (US); Darren J. Davis, Rowland Heights, CA (US); Thomas A. Soulanille, Pasadena, CA (US); Dominic Dough-Ming Cheung, South Pasadena, CA (US)

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/911,674, filed on Jul. 24, 2001, which is a continuation of application No. 09/322,677, filed on May 28, 1999, now Pat. No. 6,269,361.

Publication Classification

(51) **Int. Cl.⁷** **G06F 17/60**; G06F 7/00
(52) **U.S. Cl.** **707/1**; 707/10

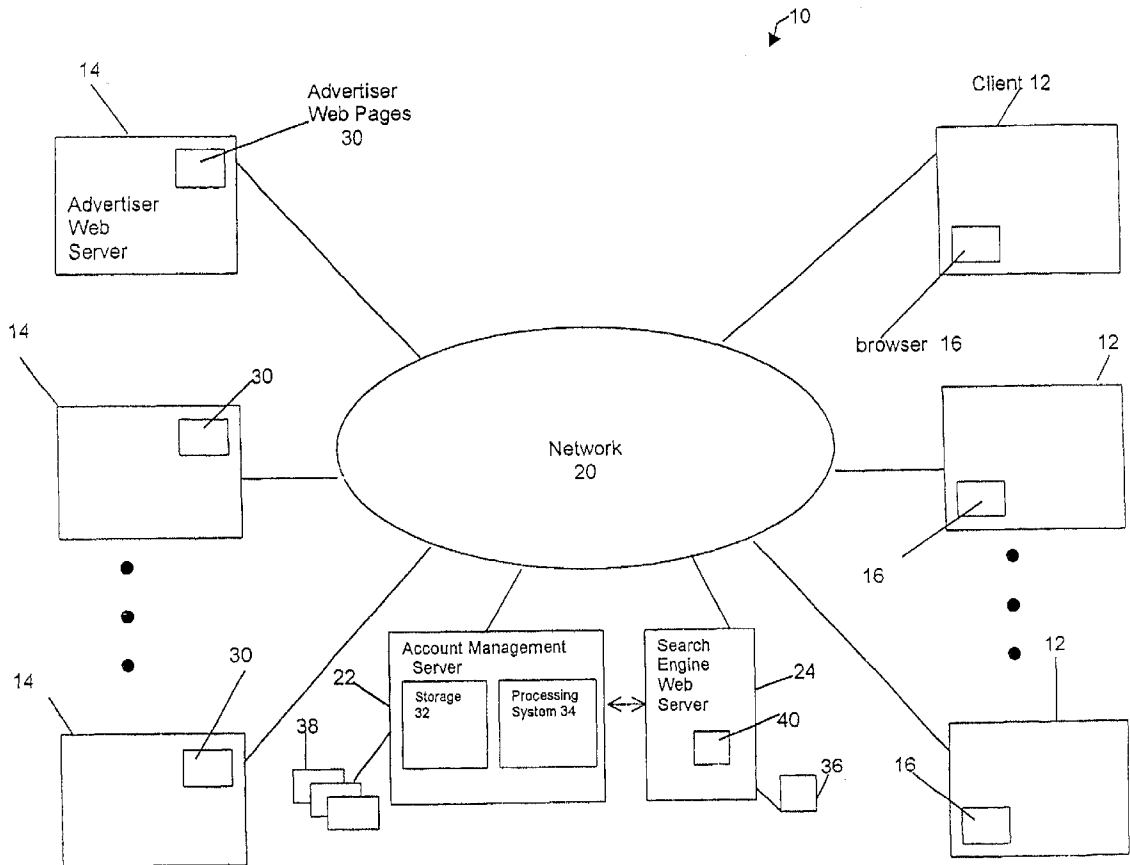
(57) **ABSTRACT**

Correspondence Address:
BRINKS HOFER GILSON & LIONE
P.O. BOX 10395
CHICAGO, IL 60610 (US)

(21) **Appl. No.: 09/963,855**

(22) **Filed: Sep. 26, 2001**

A notification method in a computer database system includes receiving a notification instruction from an owner associated with a search listing stored in the computer database system, monitoring conditions specified by the notification instruction for the search listing, and sending a notification to the owner upon detection of a changed condition of the search listing.



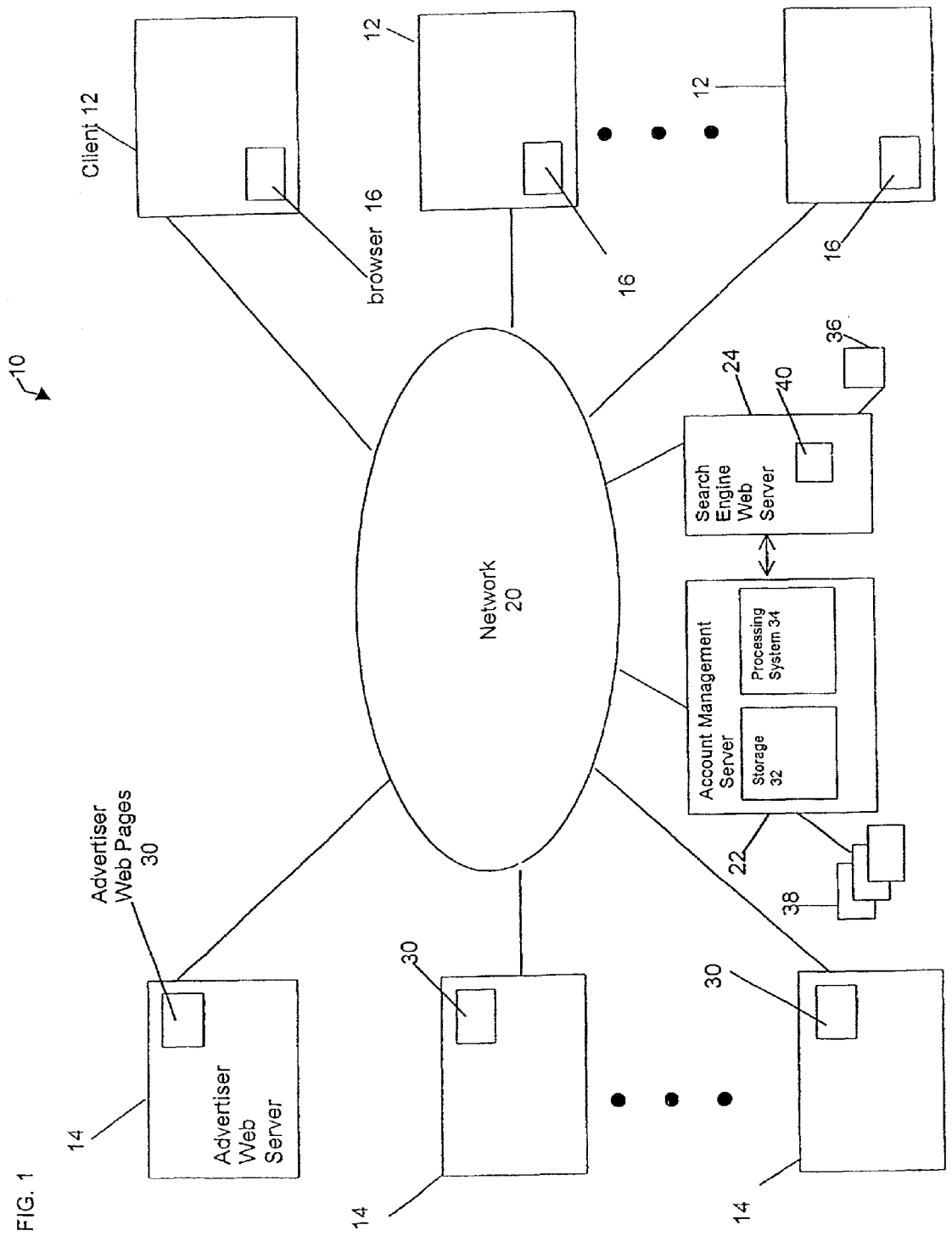


FIG. 1

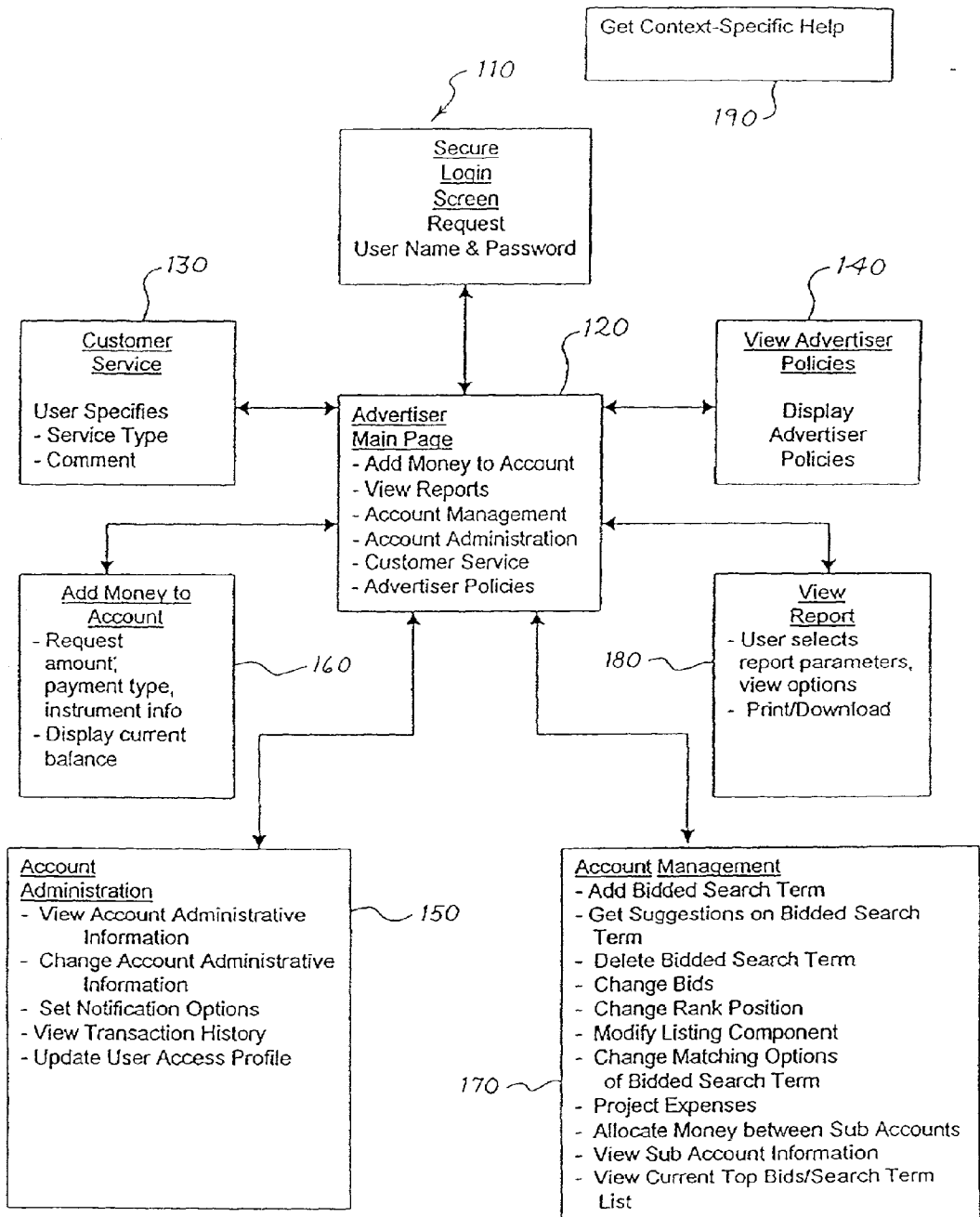


Fig. 2

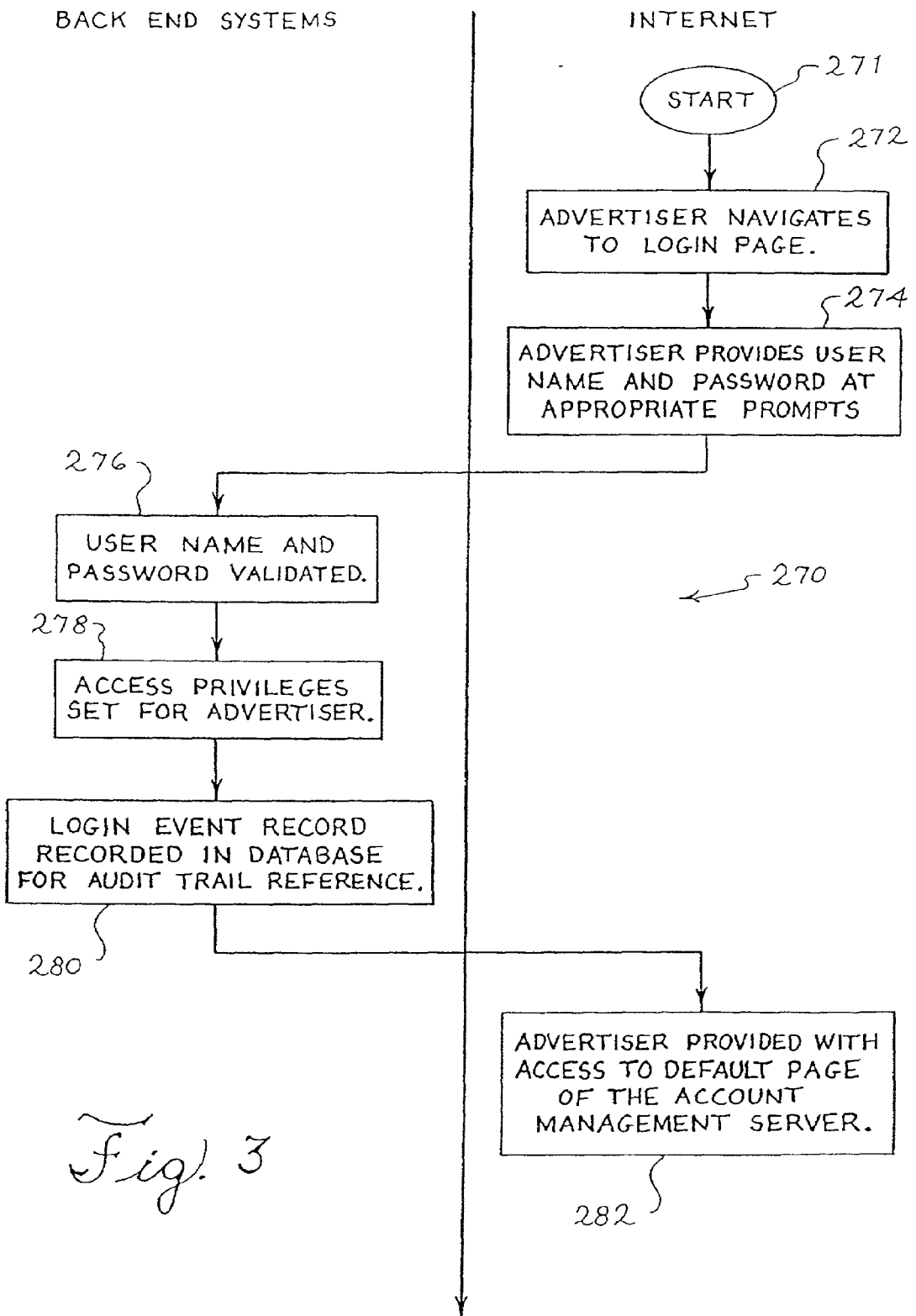


Fig. 3

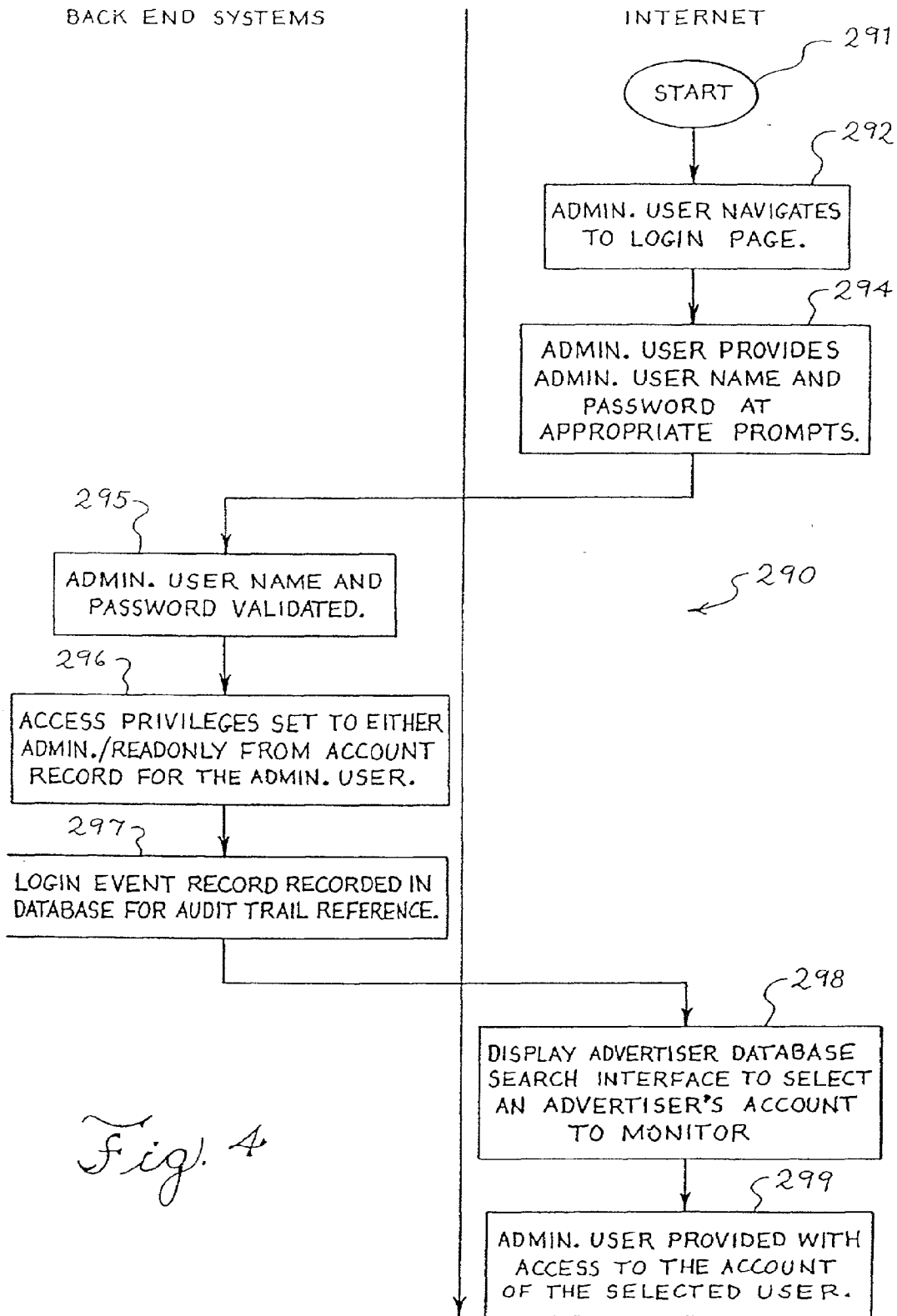


Fig. 4

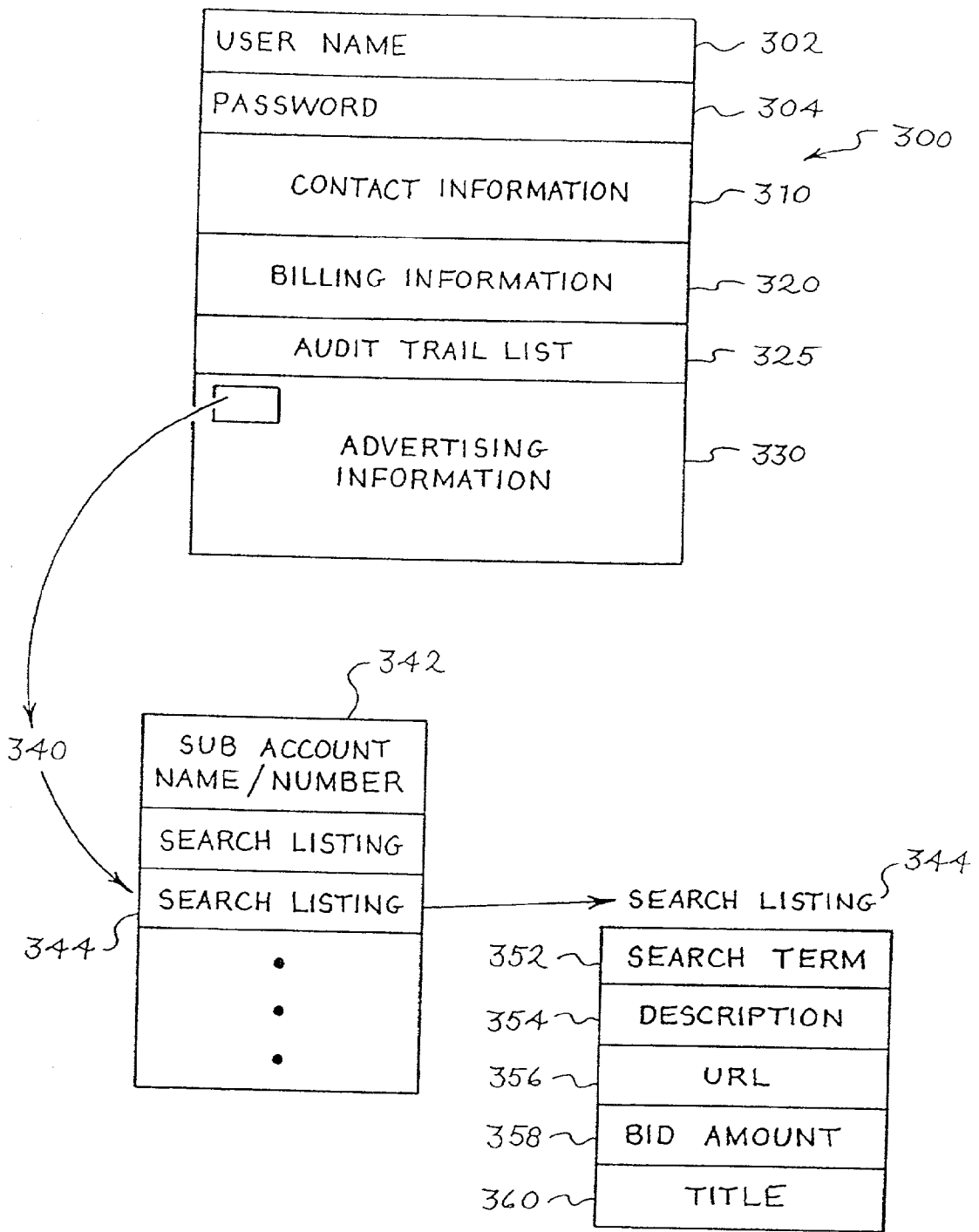


Fig. 5

Fig. 6

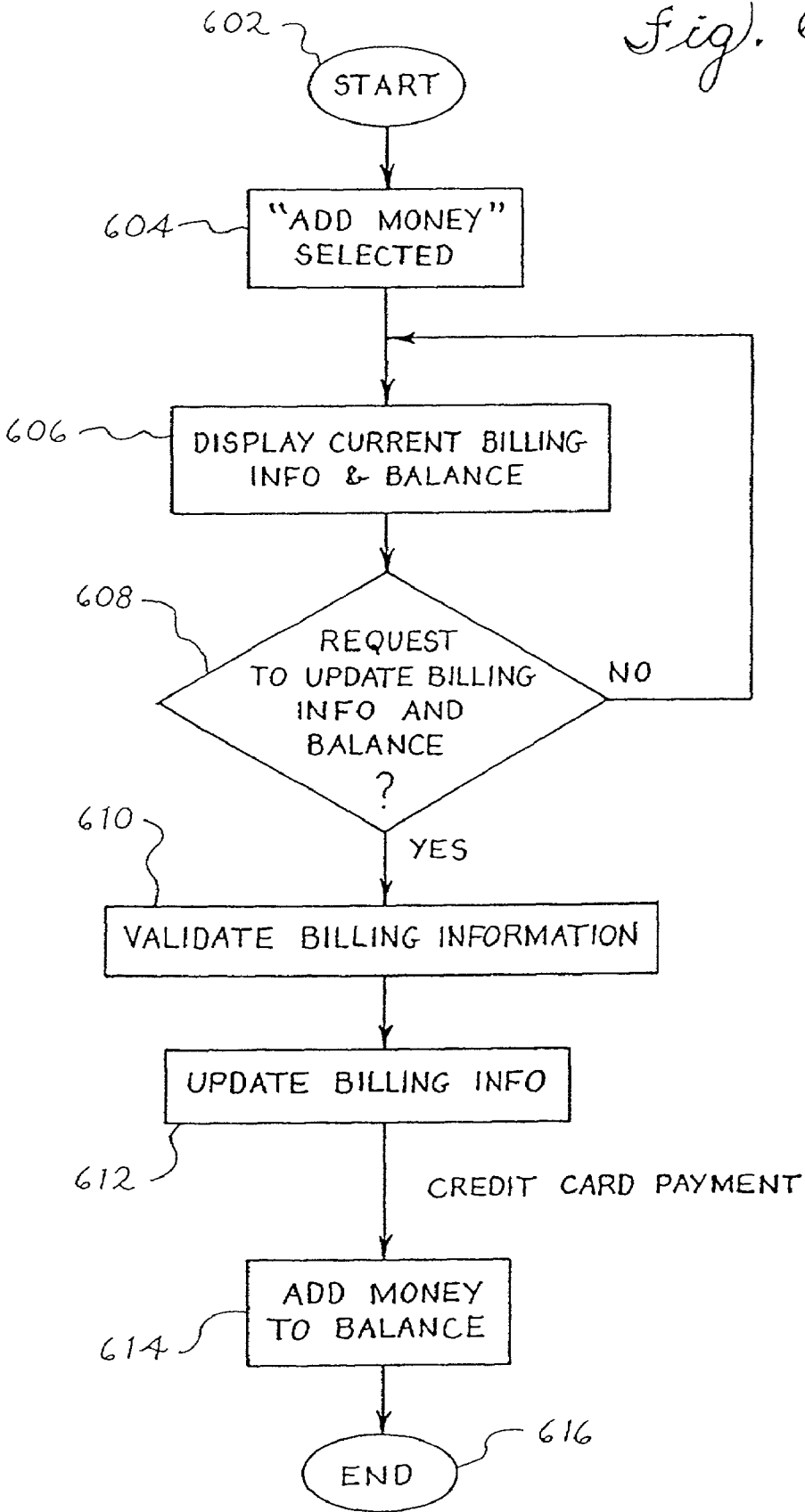
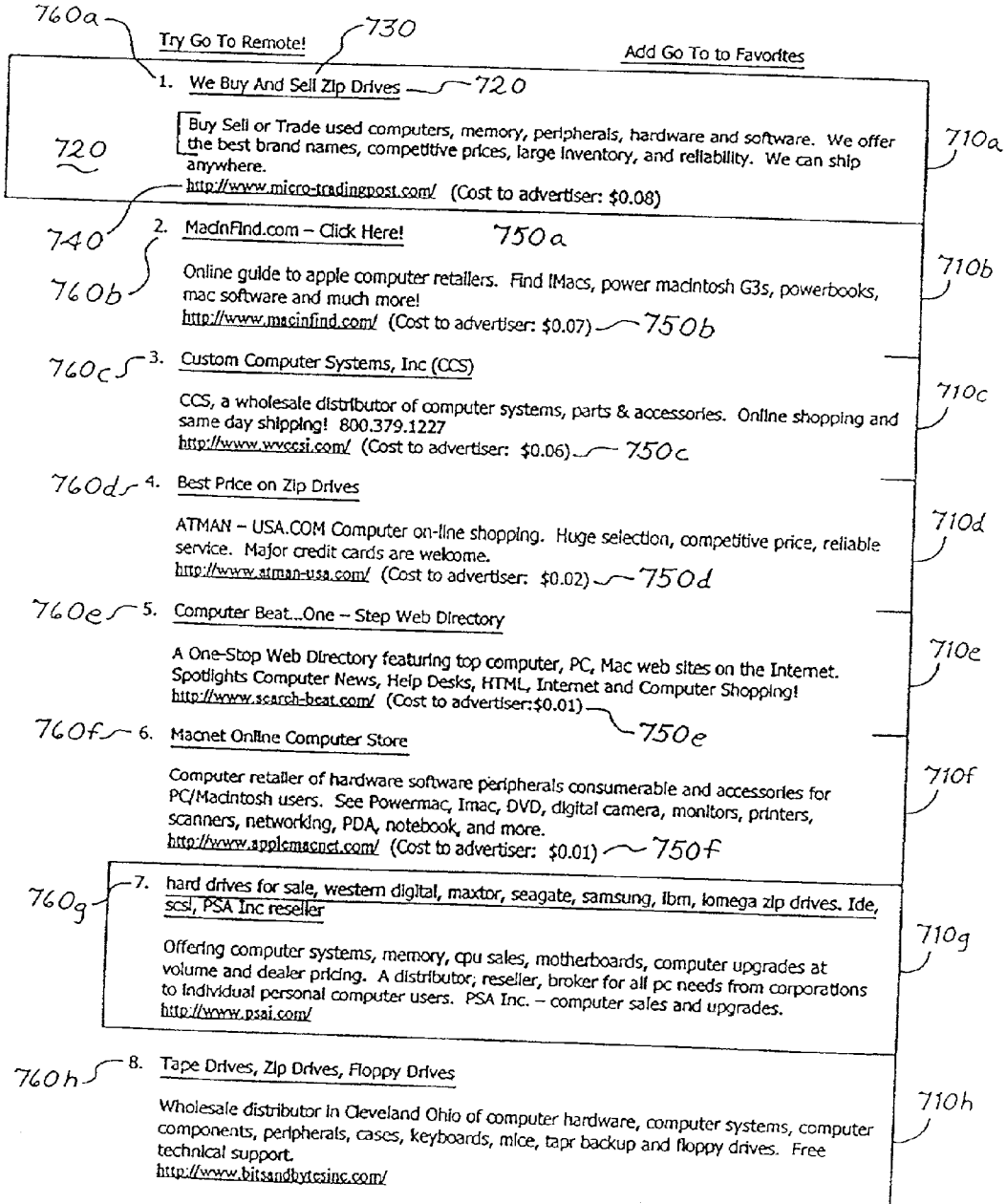
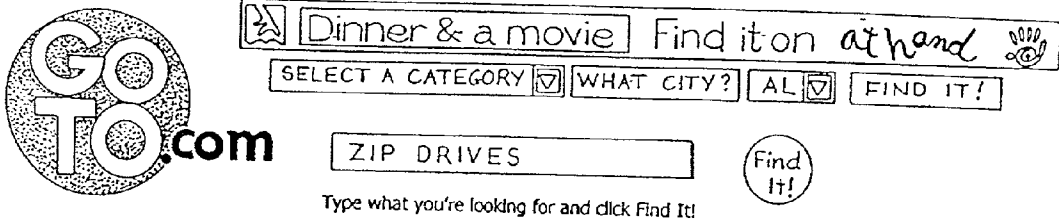


Fig. 7



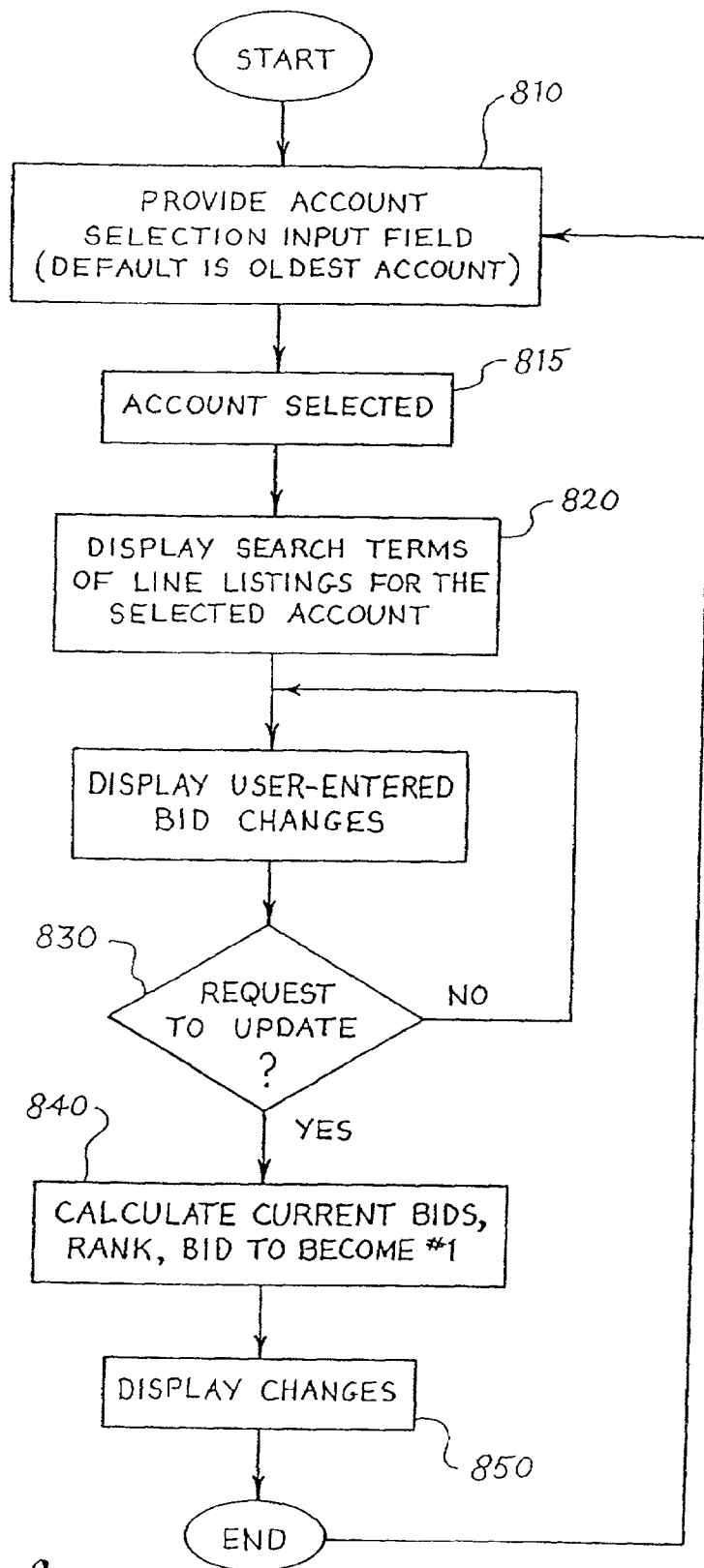


Fig. 8

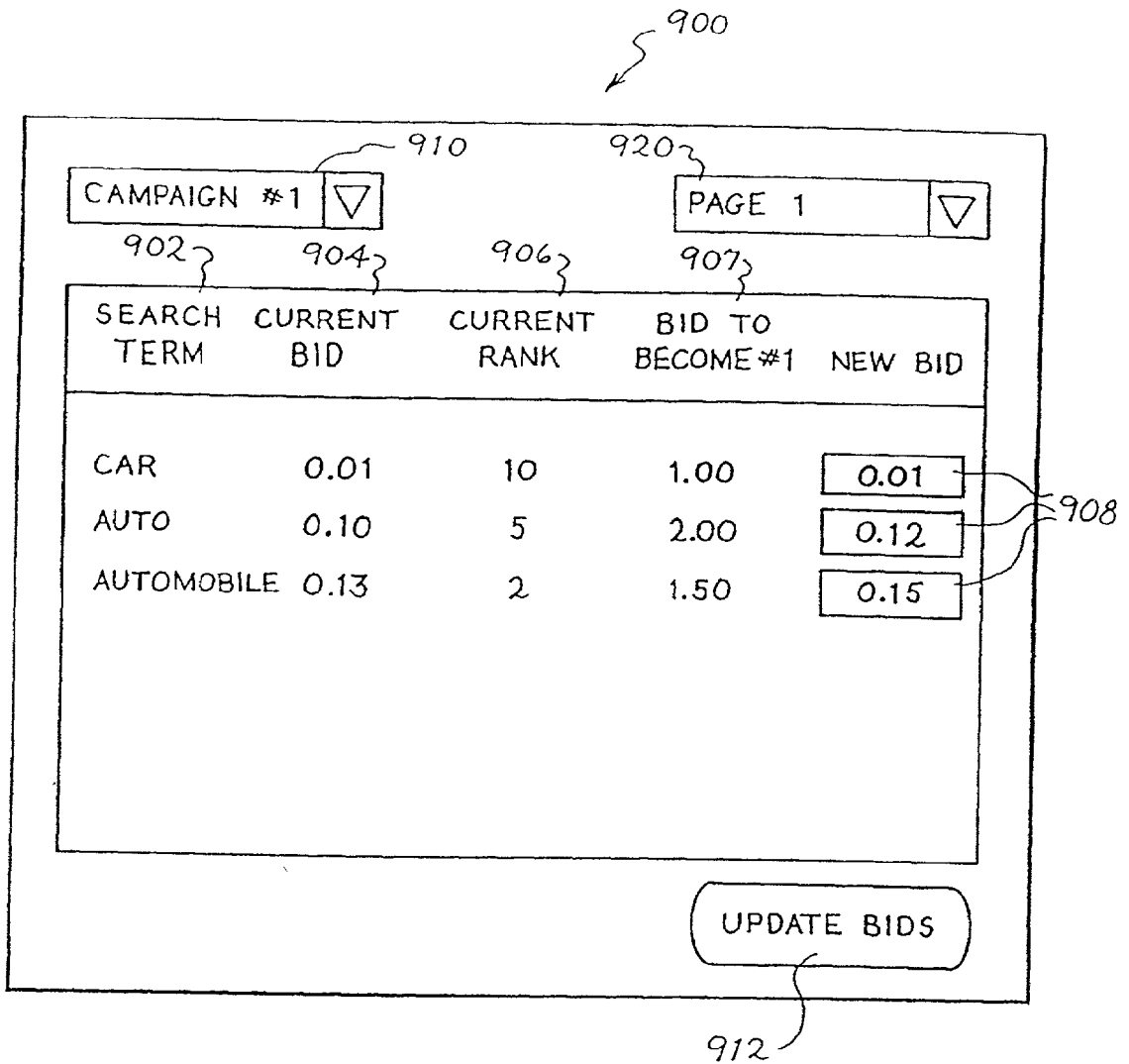


Fig. 9

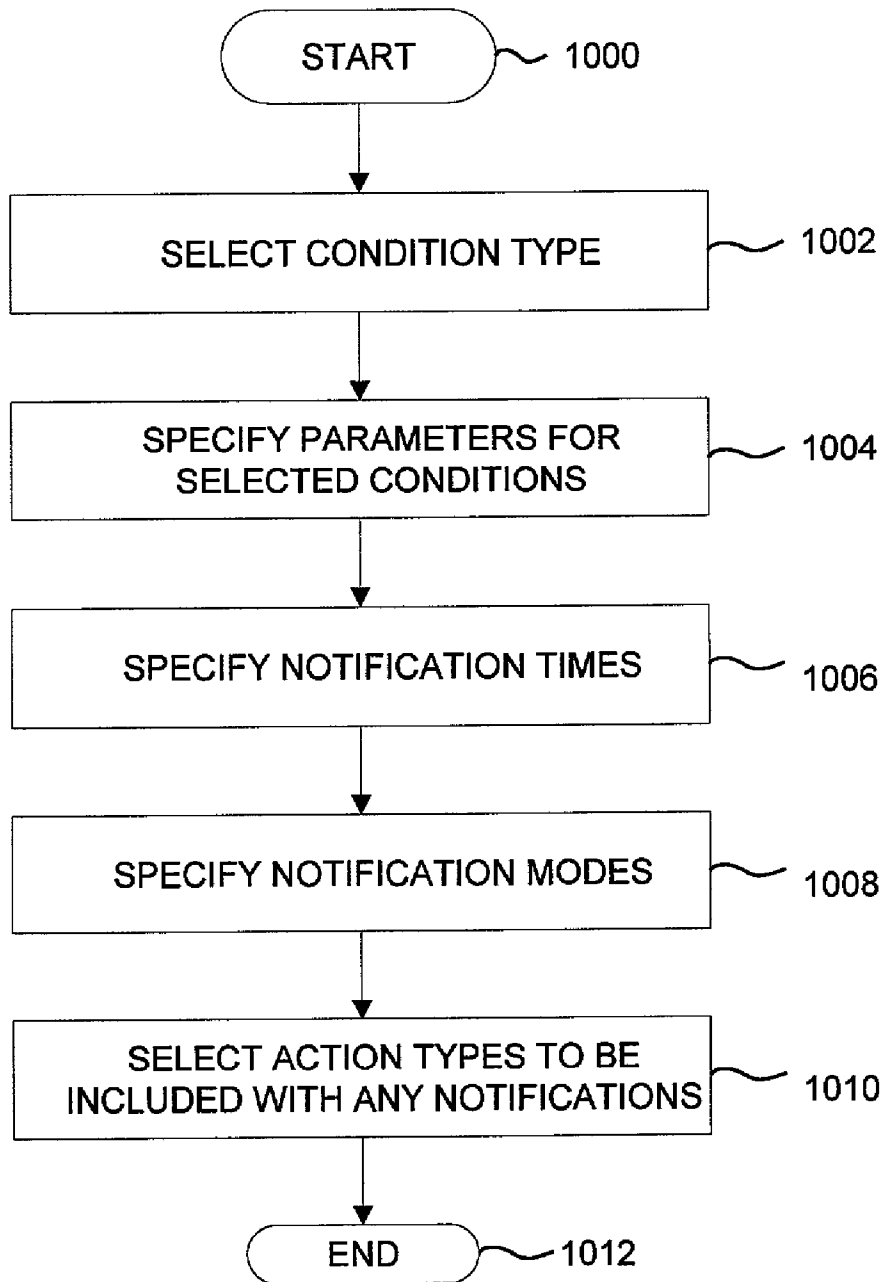


FIG. 10

FIG. 11

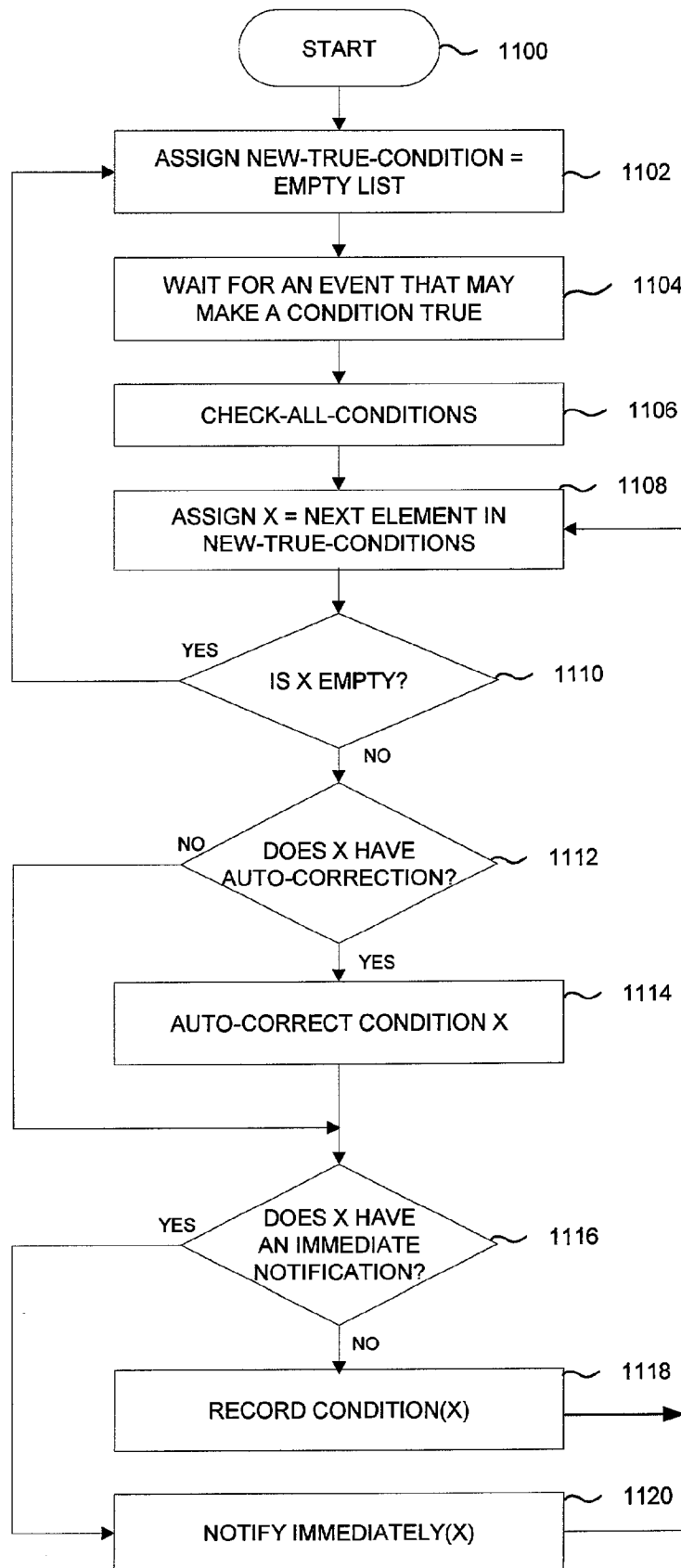


FIG. 12

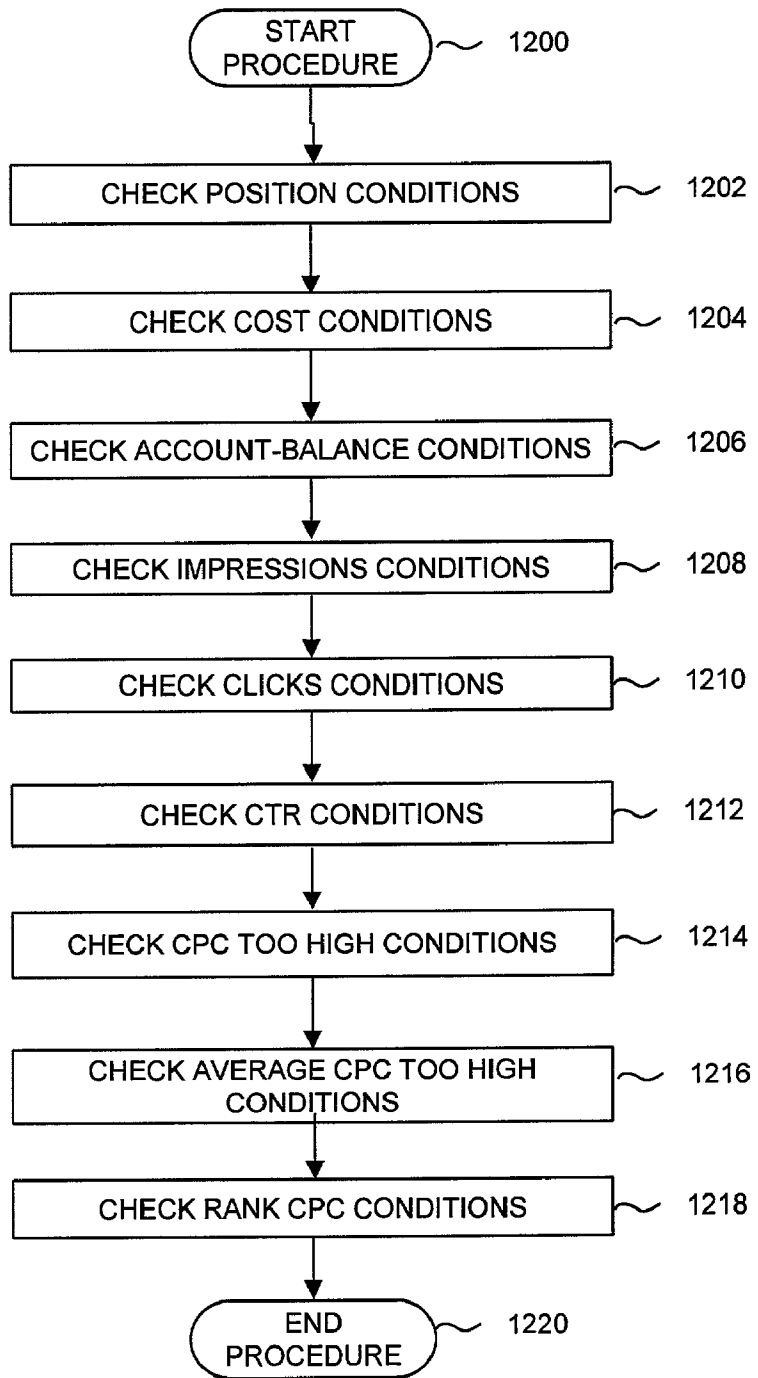


FIG. 13

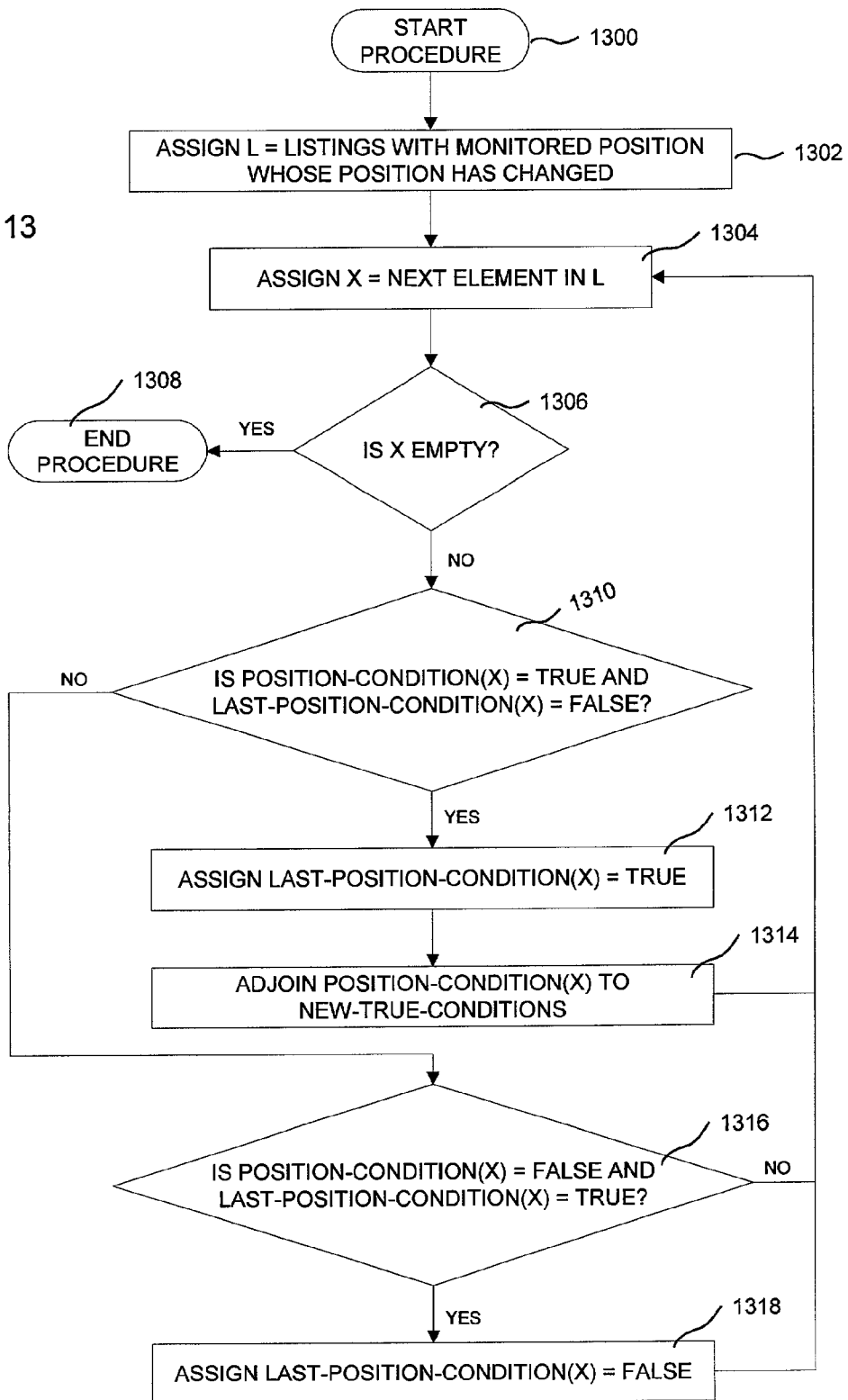


FIG. 14

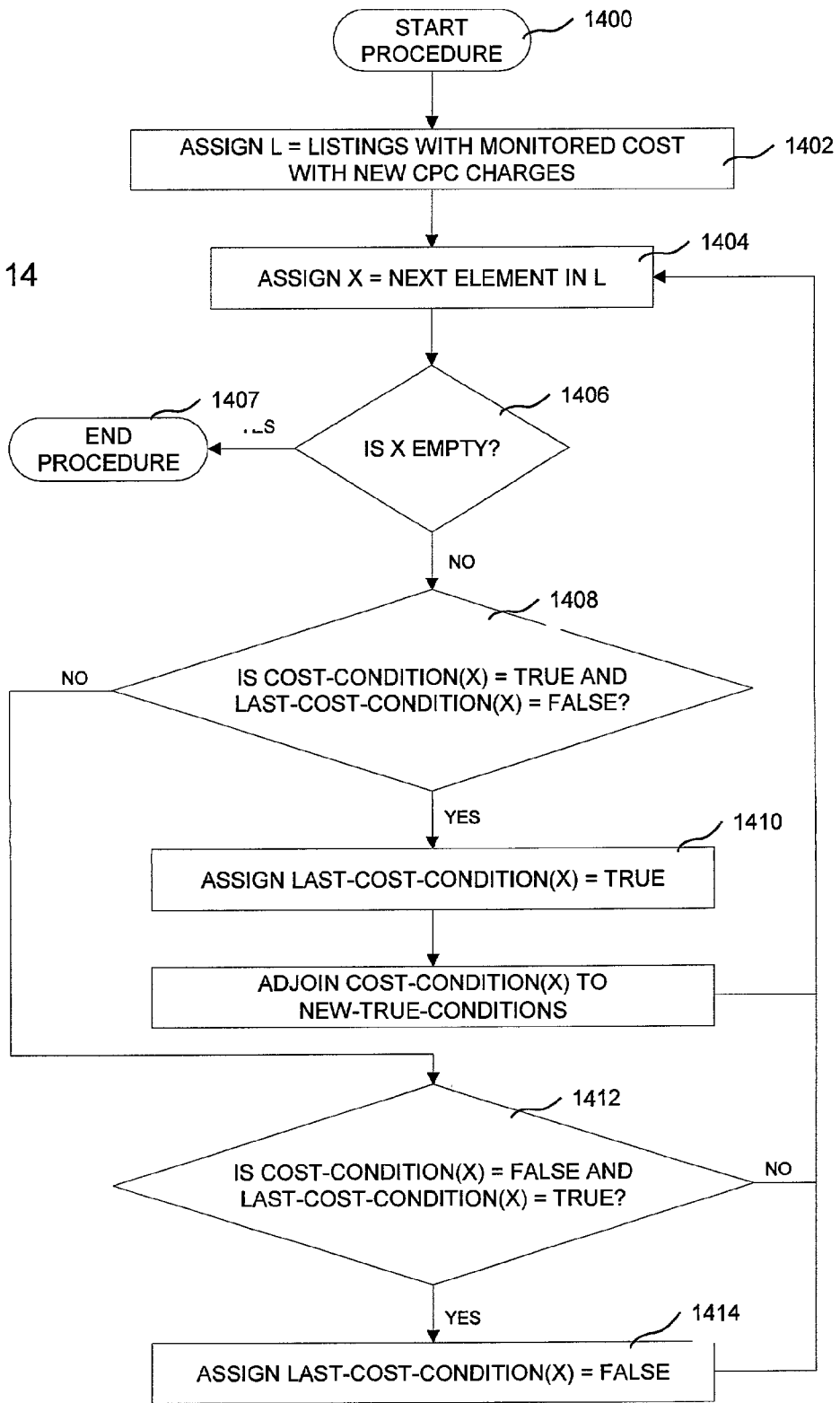


FIG. 15

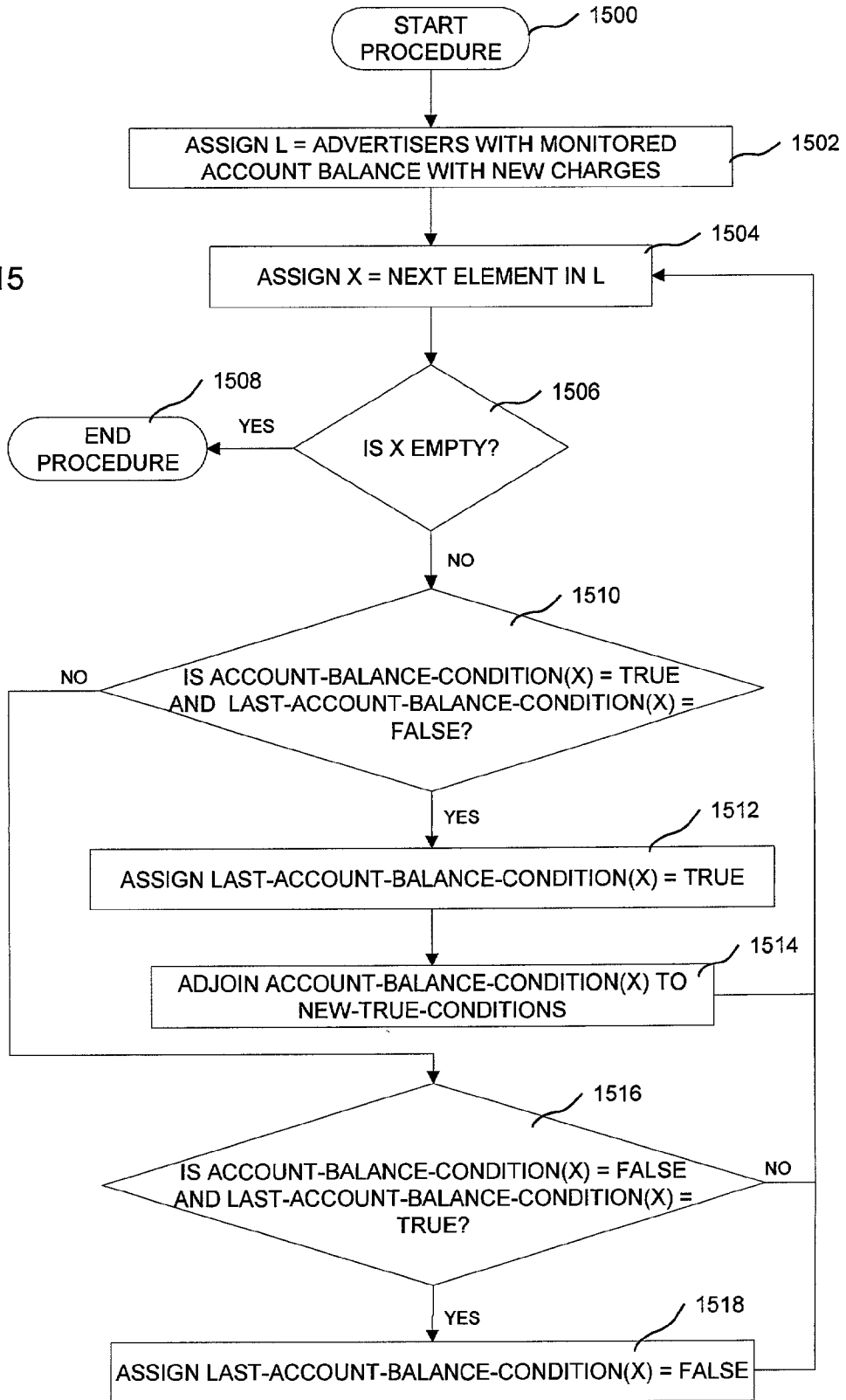


FIG. 16

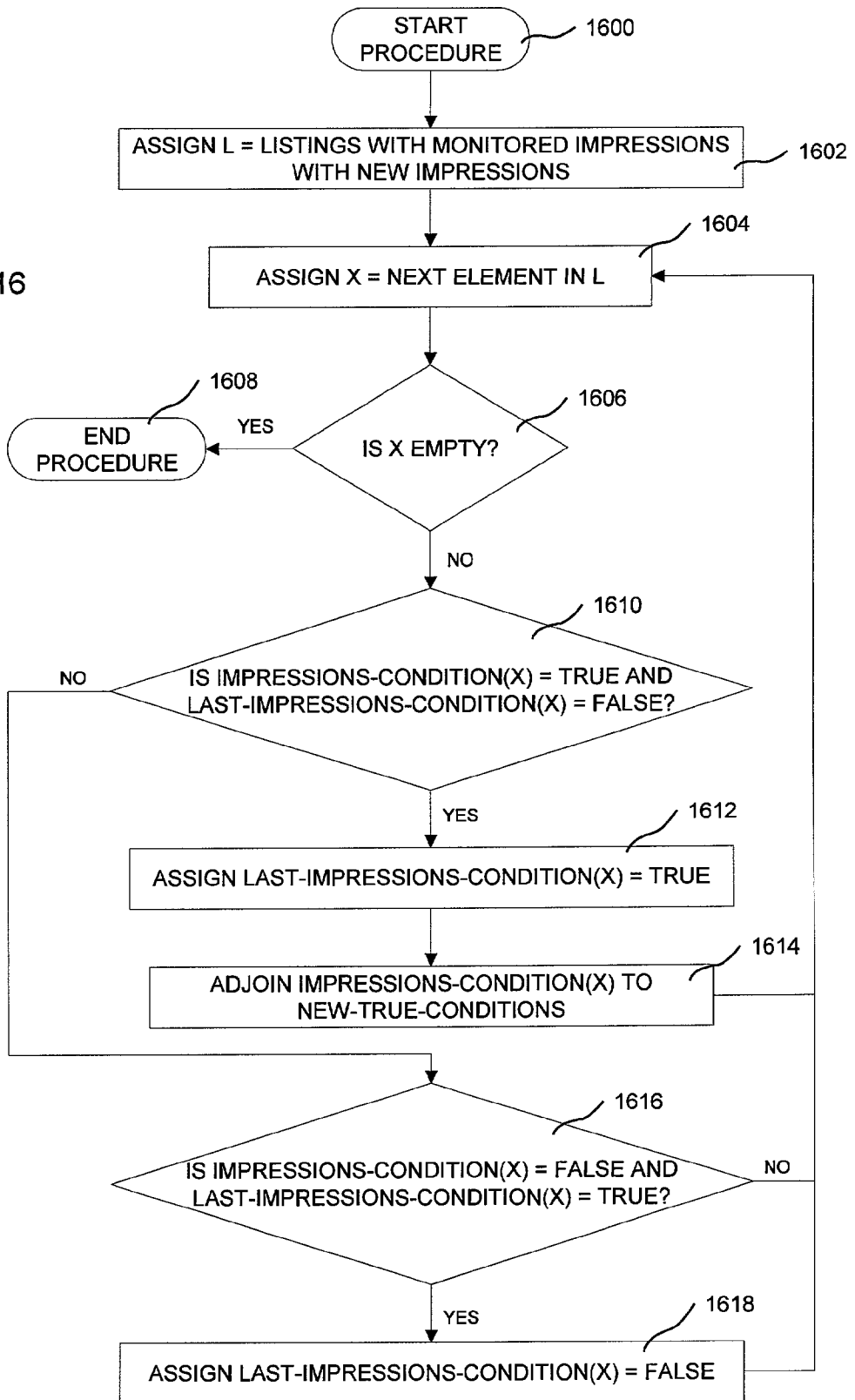


FIG. 17

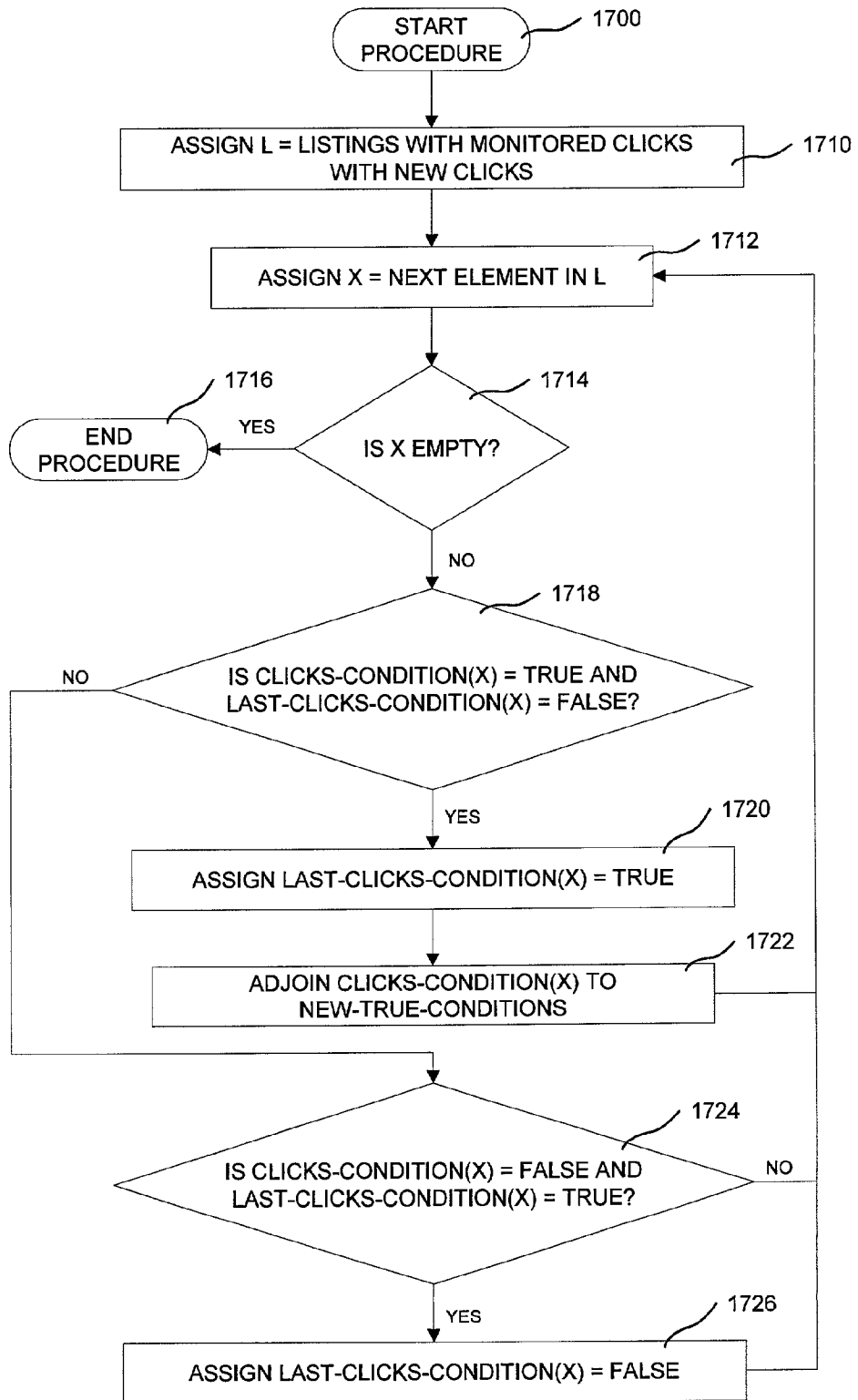


FIG. 18

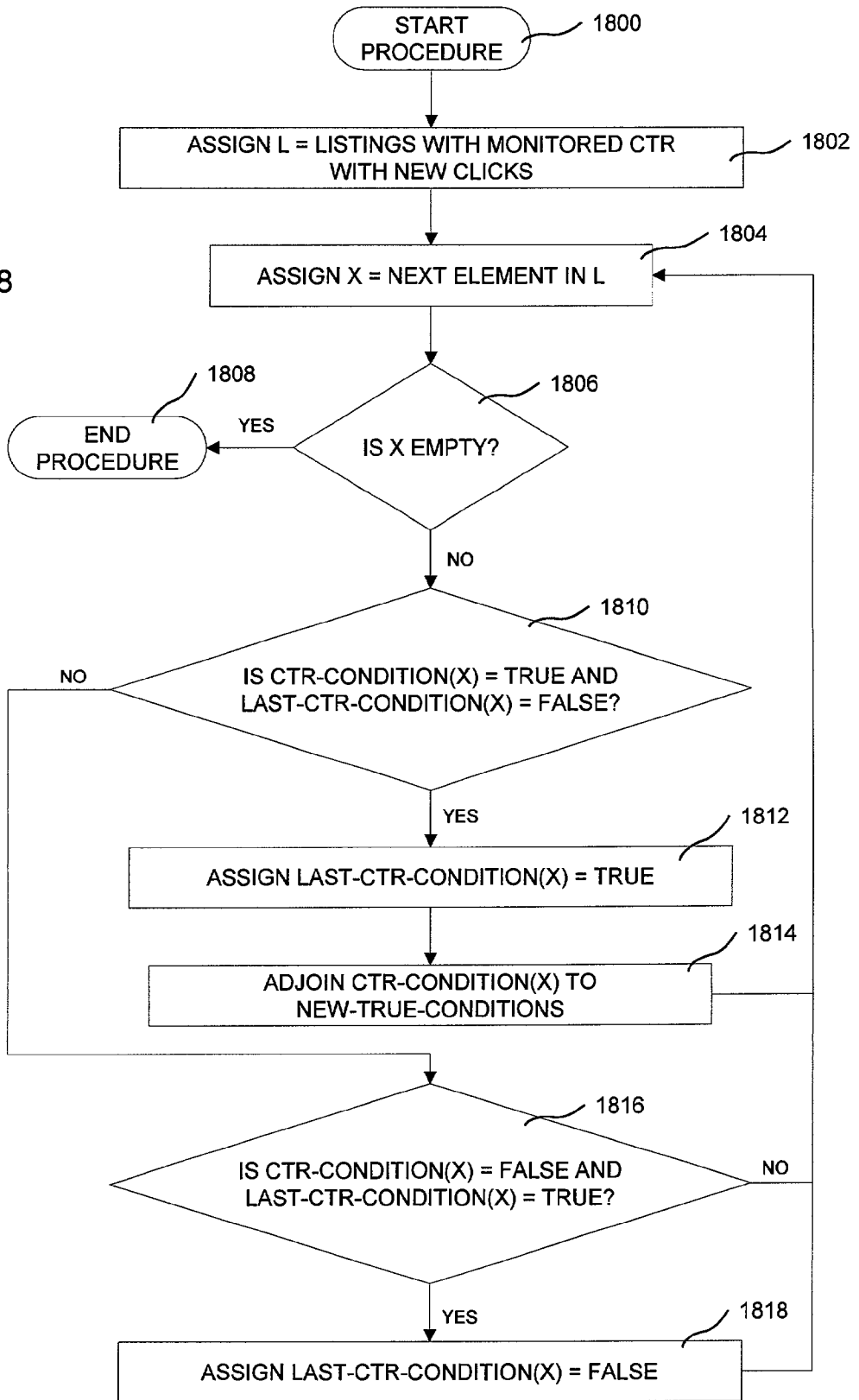


FIG. 19

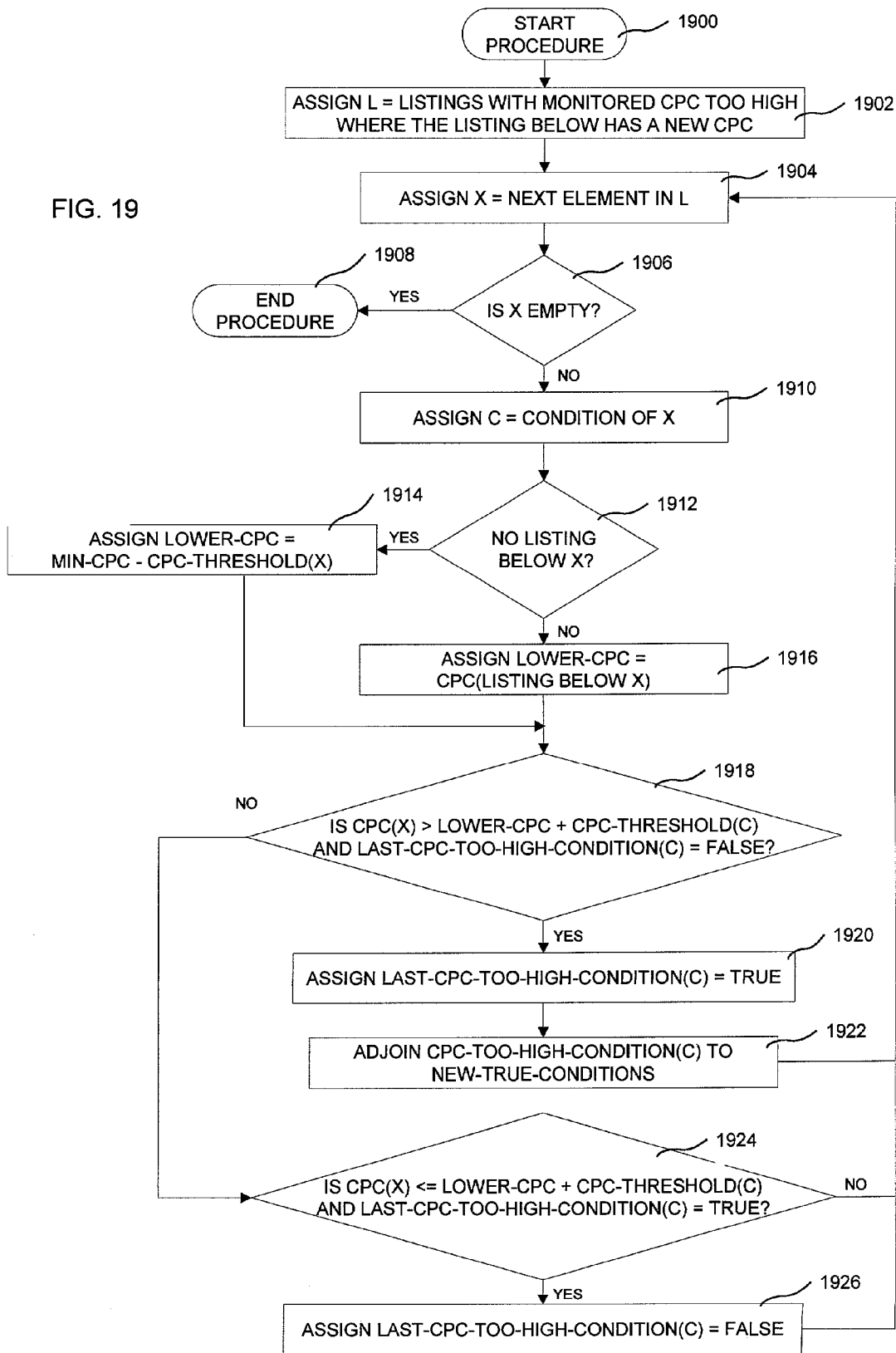
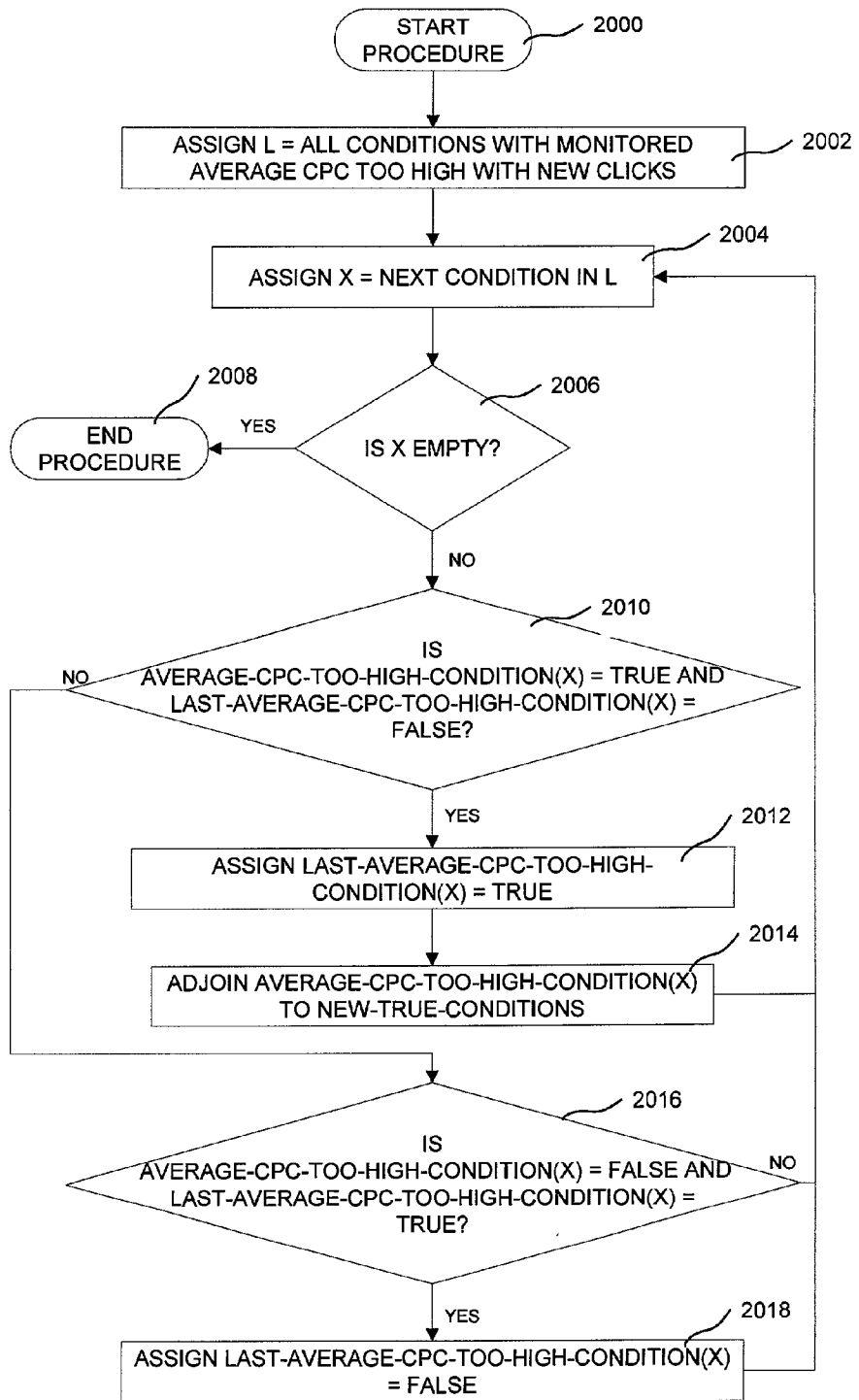


FIG. 20



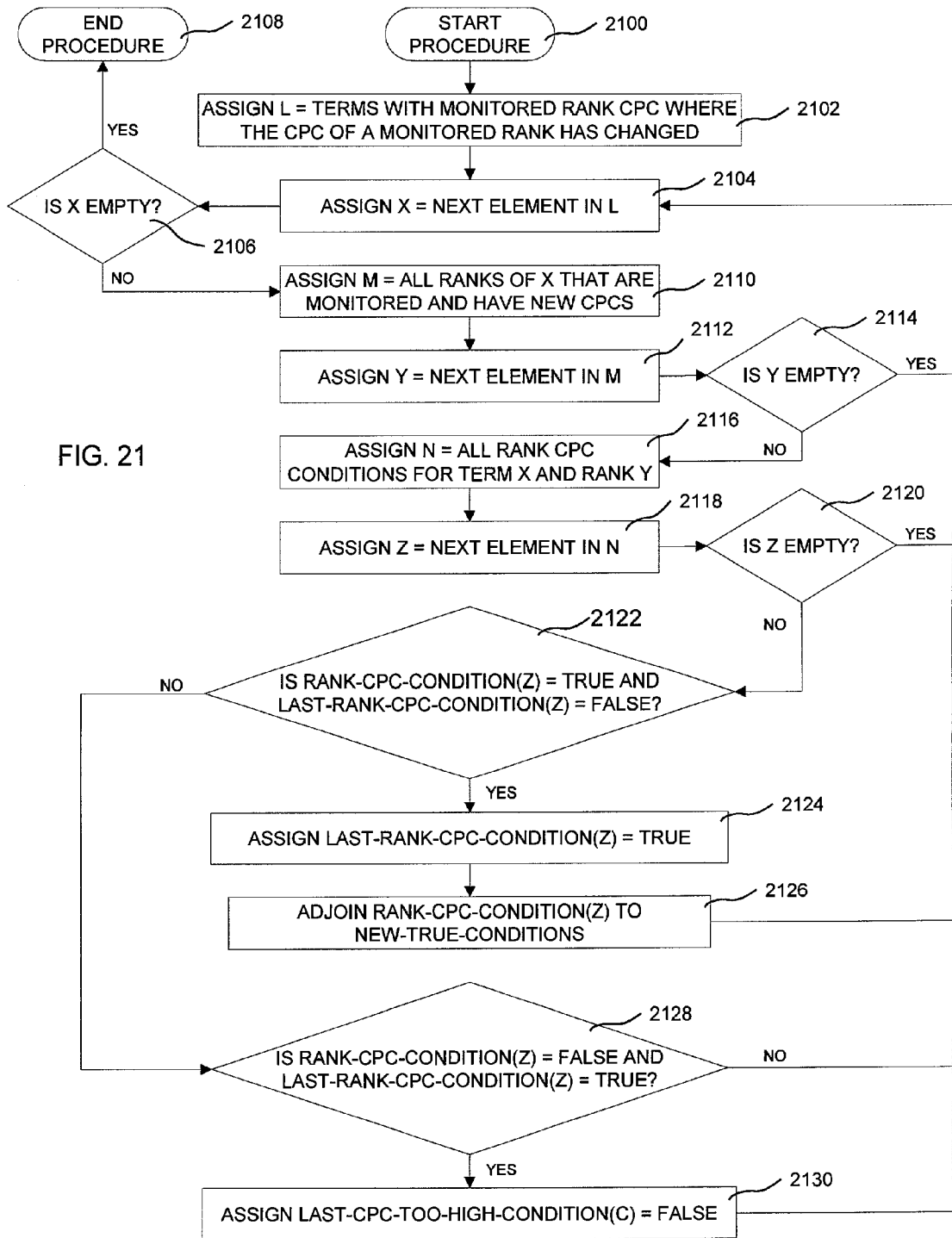
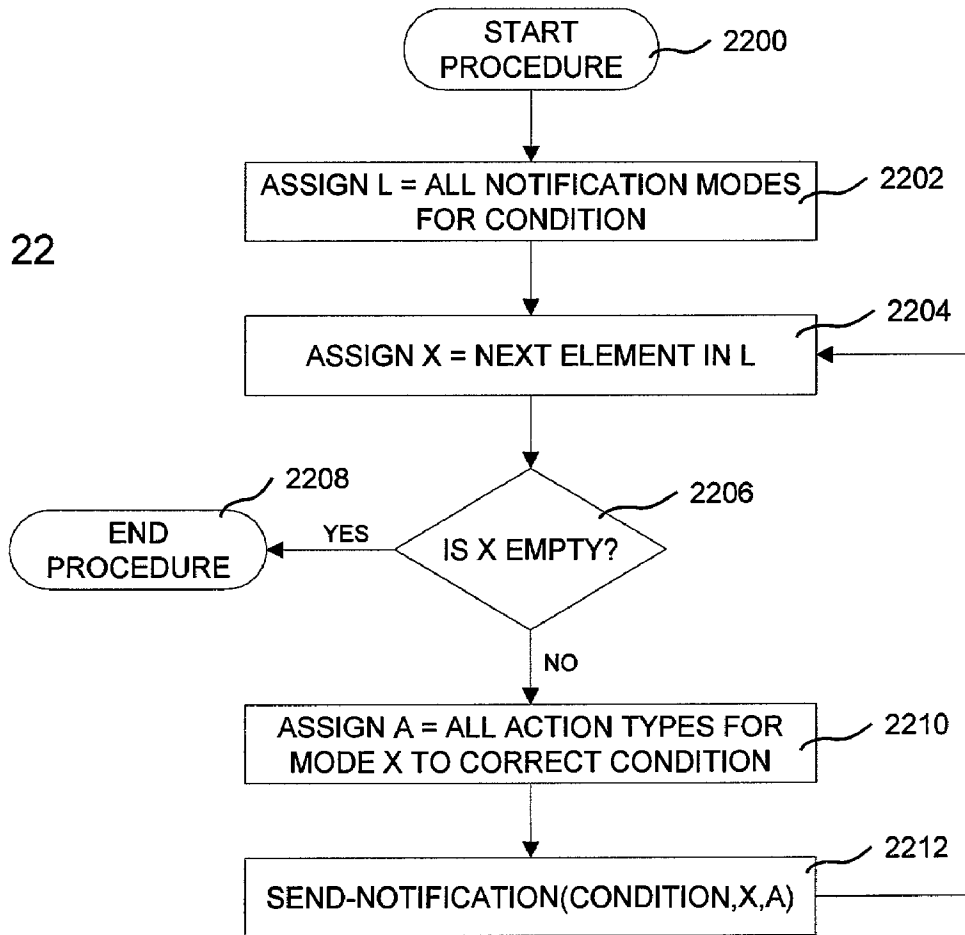


FIG. 21

FIG. 22



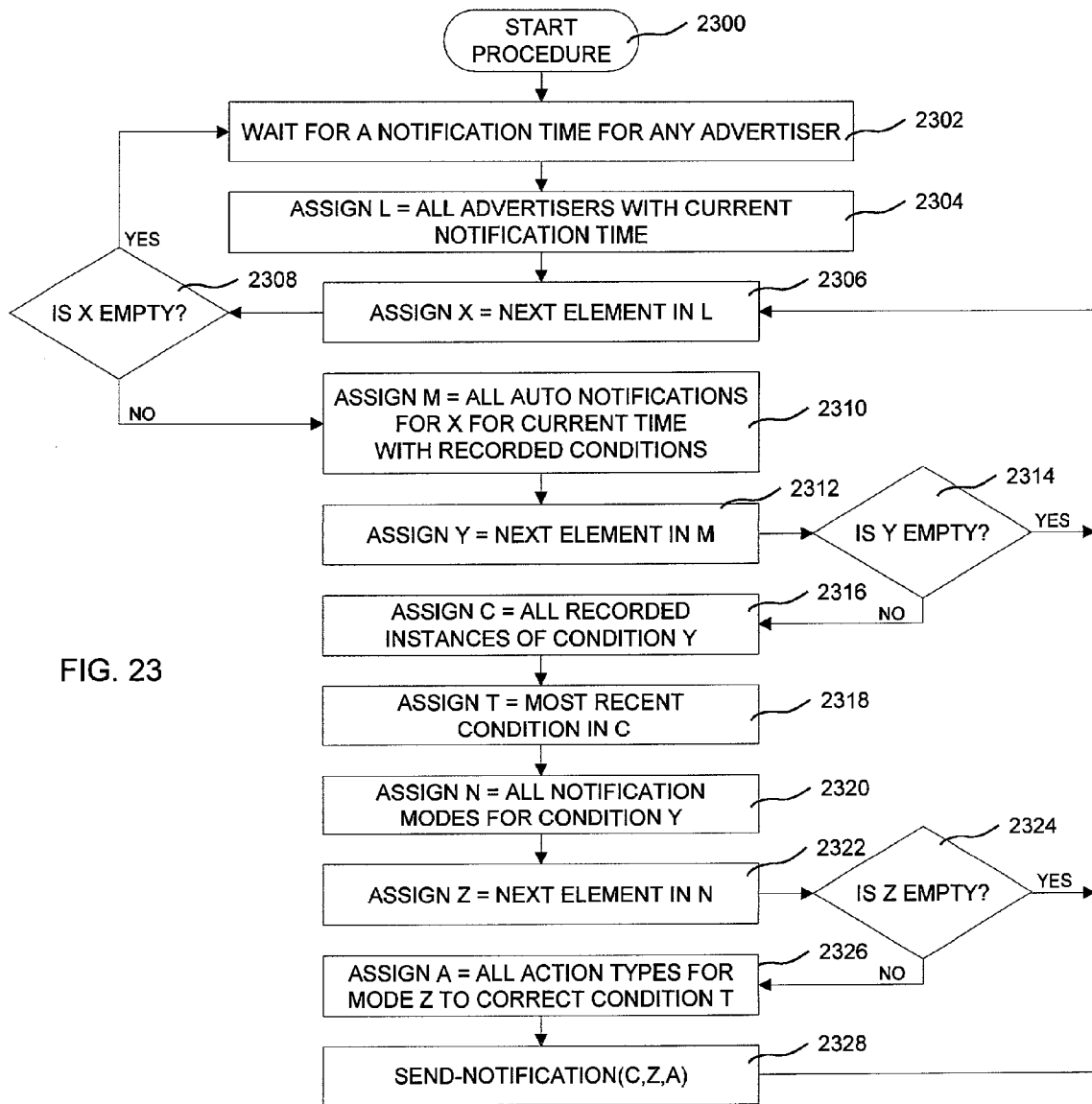
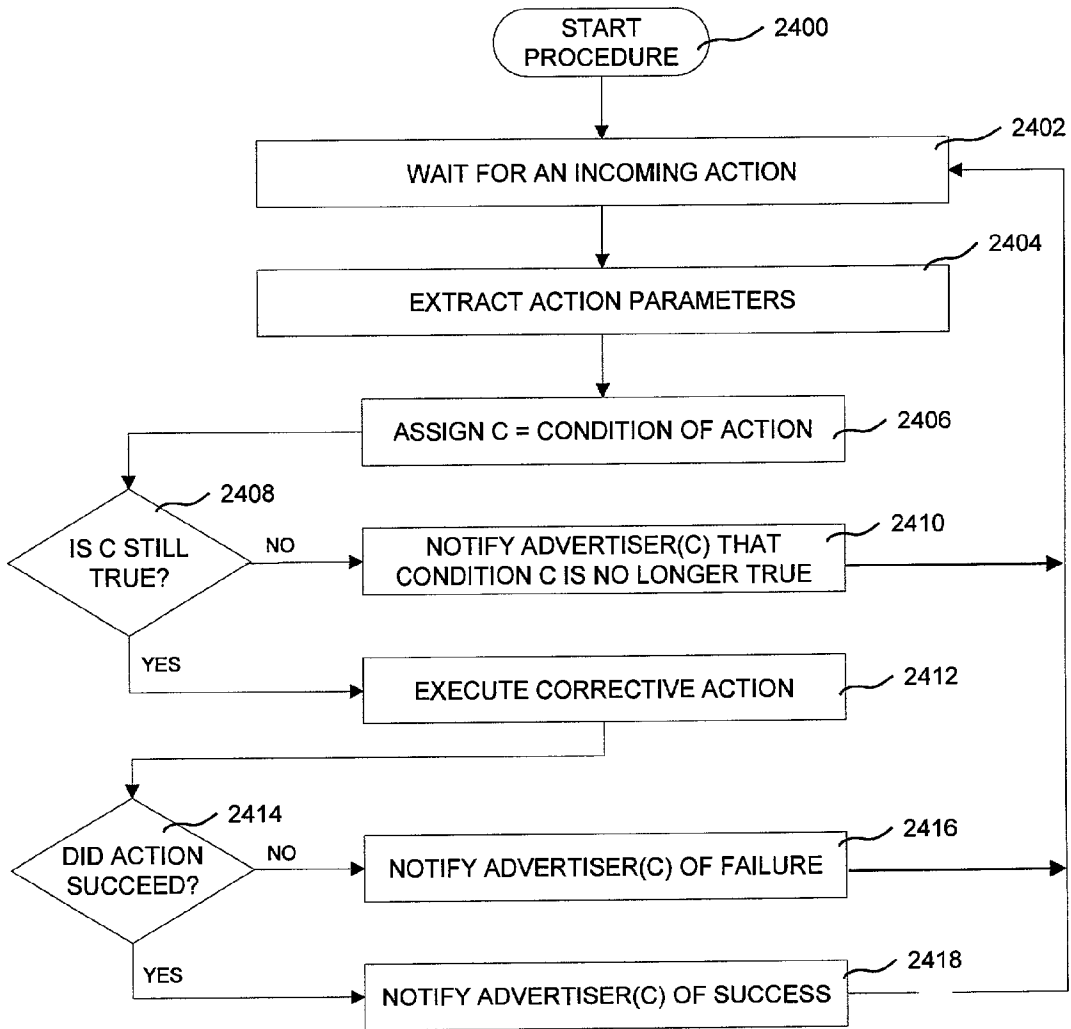


FIG. 23

FIG. 24



**AUTOMATIC ADVERTISER NOTIFICATION FOR
A SYSTEM FOR PROVIDING PLACE AND PRICE
PROTECTION IN A SEARCH RESULT LIST
GENERATED BY A COMPUTER NETWORK
SEARCH ENGINE**

**CROSS REFERENCE TO RELATED
APPLICATIONS**

[0001] This application is a continuation in part of application Ser. No. 09/911,674, filed Jul. 24, 2001 in the names of Darren J. Davis, et al., which application is incorporated herein in its entirety and which is a continuation of application Ser. No. 09/322,677, filed May 28, 1999, in the names of Darren J. Davis, et al., now U.S. Pat. No. 6,269,361, which application is also incorporated herein in its entirety.

**REFERENCE TO COMPUTER PROGRAM
LISTINGS SUBMITTED ON COMPACT DISK**

[0002] A compact disc appendix is included containing computer program code listings pursuant to 37 C.F.R. 1.52(e) and is hereby incorporated by reference in its entirety. The total number of compact discs is 1 including 24,443 files and 105,738,488 bytes. The files included on the compact disc are listed in a file entitled "dir_s" on the compact disc. Because of the large number of files contained on the compact disc, the required listing of file names, dates of creation and sizes in bytes is included in the file dir_s on the compact disk and incorporated by reference herein.

BACKGROUND OF THE INVENTION

[0003] The transfer of information over computer networks has become an increasingly important means by which institutions, corporations, and individuals do business. Computer networks have grown over the years from independent and isolated entities established to serve the needs of a single group into vast internets which interconnect disparate physical networks and allow them to function as a coordinated system. Currently, the largest computer network in existence is the Internet. The Internet is a worldwide interconnection of computer networks that communicate using a common protocol. Millions of computers, from low end personal computers to high end super computers, are connected to the Internet.

[0004] The Internet has emerged as a large community of electronically connected users located around the world who readily and regularly exchange significant amounts of information. The Internet continues to serve its original purposes of providing for access to and exchange of information among government agencies, laboratories, and universities for research and education. In addition, the Internet has evolved to serve a variety of interests and forums that extend beyond its original goals. In particular, the Internet is rapidly transforming into a global electronic marketplace of goods and services as well as of ideas and information.

[0005] This transformation of the Internet into a global marketplace was driven in large part by the introduction of an information system known as the World Wide Web ("the web"). The web is a unique distributed database designed to give wide access to a large universe of documents. The database records of the web are in the form of documents known as "pages". These pages reside on web servers and are accessible via the Internet. The web is therefore a vast

database of information dispersed across countless individual computer systems that is constantly changing and has no recognizable organization or morphology. Computers connected to the Internet may access the web pages via a program known as a browser, which has a powerful, simple-to-learn graphical user interface. One powerful technique supported by the web browser is known as hyperlinking, which permits web page authors to create links to other web pages which users can then retrieve by using simple point-and-click commands on the web browser.

[0006] The pages may be constructed in any one of a variety of formatting conventions, such as Hyper Text Markup Language (HTML), and may include multimedia information content such as graphics, audio, and moving pictures. Any person with a computer and a connection to the Internet may access any publicly accessible page posted on the web. Thus, a presence on the World Wide Web has the capability to introduce a worldwide base of consumers to businesses, individuals, and institutions seeking to advertise their products and services to potential customers. Furthermore, the ever increasing sophistication in the design of web pages, made possible by the exponential increase in data transmission rates and computer processing speeds, makes the web an increasingly attractive medium for advertising and other business purposes, as well as for the free flow of information.

[0007] The availability of powerful new tools that facilitate the development and distribution of Internet content has led to a proliferation of information, products, and services offered on the Internet and dramatic growth in the number of consumers using the Internet. International Data Corporation, commonly referred to as IDC, estimates that the number of Internet users will grow from approximately 97 million worldwide in 1998 to approximately 320 million worldwide by the end of 2002. In addition, commerce conducted over the Internet has grown and is expected to grow dramatically. IDC estimates that the percentage of Internet users buying goods and services on the Internet will increase from approximately 28% at the end of 1998 to approximately 40% in 2002, and that over the same period of time, the total value of goods and services purchased over the Internet will increase from approximately \$32.4 billion to approximately \$425.7 billion.

[0008] The Internet has emerged as an attractive new medium for advertisers of information, products and services to reach consumers. However, the World Wide Web is composed of a seemingly limitless number of web pages dispersed across millions of different computer systems all over the world in no discernible organization. Mechanisms, such as directories and search engines, have been developed to index and search the information available on the web and thereby help Internet users locate information of interest. These search services enable consumers to search the Internet for a listing of web sites based on a specific topic, product, or service of interest.

[0009] Search services are, after e-mail, the most frequently used tool on the Internet. As a result, web sites providing search services have offered advertisers significant reach into the Internet audience and have given advertisers the opportunity to target consumer interests based on keyword or topical search requests.

[0010] In a web-based search on an Internet search engine, a user enters a search term comprising one or more key-

words, which the search engine then uses to generate, in real time, a listing of web pages that the user may access via a hyperlink. The search engines and web site directories of the prior art, however, rely upon processes for assigning results to keywords that often generate irrelevant search results. The automated search technology that drives many search engines in the prior art rely in large part on complex, mathematics-based database search algorithms that select and rank web pages based on multiple criteria such as keyword density and keyword location. The search results generated by such mechanisms often rely on blind mathematical formulas and may be random and even irrelevant. In addition, search engines that use automated search technology to catalog search results generally rely on invisible web site descriptions, or "meta tags", that are authored by web site promoters. Web site owners may freely tag their sites as they choose. Consequently, some web site promoters or promoters insert popular search terms into their web site meta tags which are not relevant because by doing so they may attract additional consumer attention at little to no marginal cost. Finally, many web sites have similar meta tags, and the search engines of the prior art are simply not equipped to prioritize results in accordance with consumers' preferences.

[0011] Search engines and web site directories may also rely on the manual efforts of limited editorial staffs to review web page information. Since comprehensive manual review and indexing of an unpredictable, randomly updated database such as the web is an impossible task, search engine results are often incomplete or out-of-date. Moreover, as the volume and diversity of Internet content has grown, on many popular web search sites, consumers must frequently click-through multiple branches of a hierarchical directory to locate web sites responsive to their search request, a process that is slow and unwieldy from the consumer's standpoint. Thus, the prior art search engines are ineffective for web page owners seeking to target their web exposure and distribute information to the attention of interested users on a current and comprehensive basis.

[0012] Furthermore, current paradigms for generating web site traffic, such as banner advertising, follow traditional advertising paradigms and fail to utilize the unique attributes of the Internet. In the banner advertising model, web site promoters seeking to promote and increase their web exposure often purchase space on the pages of popular commercial web sites. The web site promoters usually fill this space with a colorful graphic, known as a banner, advertising their own web site. The banner may act a hyperlink a visitor may click on to access the site. Like traditional advertising, banner advertising on the Internet is typically priced on an impression basis with advertisers paying for exposures to potential consumers. Banners may be displayed at every page access, or, on search engines, may be targeted to search terms.

[0013] Nonetheless, impression-based advertising inefficiently exploits the Internet's direct marketing potential, as the click-through rate, the rate of consumer visits a banner generates to the destination site, may be quite low. Web site promoters are therefore paying for exposure to many consumers who are not interested in the product or service being promoted, as most visitors to a web site seek specific information and may not be interested in the information announced in the banner. Likewise, the banner often fails to

reach interested individuals, since the banner is not generally searchable by search engines and the interested persons may not know where on the web to view the banner.

[0014] Thus, the traditional paradigms of advertising and search engine algorithms fail to effectively deliver relevant information via the World Wide Web to interested parties in a cost-effective manner. Internet advertising can offer a level of targetability, interactivity, and measurability not generally available in other media. With the proper tools, Internet advertisers have the ability to target their messages to specific groups of consumers and receive prompt feedback as to the effectiveness of their advertising campaigns.

[0015] Ideally, web site promoters should be able to control their placement in search result listings so that their listings are prominent in searches that are relevant to the content of their web site. The search engine functionality of the Internet needs to be focused in a new direction to facilitate an on-line marketplace which offers consumers quick, easy and relevant search results while providing Internet advertisers and promoters with a cost-effective way to target consumers. A consumer utilizing a search engine that facilitates this on-line marketplace will find companies or businesses that offer the products, services, or information that the consumer is seeking. In this on-line marketplace, companies selling products, services, or information bid in an open auction environment for positions on a search result list generated by an Internet search engine. Since advertisers must pay for each click-through referral generated through the search result lists generated by the search engine, advertisers have an incentive to select and bid on those search keywords that are most relevant to their web site offerings. The higher an advertiser's position on a search result list, the higher likelihood of a "referral"; that is, the higher the likelihood that a consumer will be referred to the advertiser's web site through the search result list. The openness of this advertising marketplace is further facilitated by publicly displaying, to consumers and other advertisers, the price bid by an advertiser on a particular search result listing.

[0016] U.S. Pat. No. 6,269,361 describes a system and method for enabling promoters to influence a position on a search result listing generated by an Internet search engine for a specified set of search terms. The system and method enable advertisers to specify key search terms to the search engine so as to target their search result list placement to the search queries most relevant to their business. Further, the system and method enable promoters to examine their current search term and placement couplings online and to make substantially instantaneous changes to their selected search terms, placements, and web site titles and descriptions.

[0017] In this system, advertisers, or web site promoters, establish bid amounts for search listings with a pay for performance web site or marketplace operator which are chargeable to the advertiser by the marketplace web site operator. In response to a received query from a searcher, search listings are located, arranged according to bid and displayed to the searcher. If a searcher selects or clicks through an advertiser's search listing, the bid amount is charged to the advertiser by the pay for performance web site operator. Advertisers can control the position of their search listing in the search result list by adjusting the bid amount associated with the search listing.

[0018] The method described in the U.S. Pat. No. 6,269,361 can be burdensome to manage for an advertiser. In particular, advertisers want to maintain favorable positions in the search results (so as to obtain a high volume of qualified traffic) at a favorable price. The system described in U.S. Pat. No. 6,269,361 provides no ready means to do that. Advertisers can resort to frequent inspection of their ranking on search terms that are important to them, for example by performing a search on www.goto.com. When they observe a change as a consequence of competing advertisers' bidding activities, they can log in to the pay for performance website and change their bids manually in response. In the case where they have been outbid for a position they want to retain, they can increase their bid to retake the position, if the required cost per click ("CPC"), which is equal to the amount of their bid, is one they are willing to pay. In the case where the bid of the listing ranked below theirs has decreased, some advertisers may wish to lower their bid to reduce the amount they pay while still maintaining their position in the results set.

[0019] There are many other tasks that advertisers typically perform in addition to managing the position of their listings, including keeping track of the accumulated costs of listings, the number of clicks of listings, the click through rate (CTR) of listings, and checking their account balance. In addition, advertisers have to constantly keep track of the changing marketplace, e.g., to check if the bid of a listing is too high, or if a more desirable rank is now affordable.

[0020] Managing the budget is a vital business concern for advertisers, and there is a need to keep track of the breakdown of expenses for different terms. For example, around Father's Day, the number of searches for the term "tie" may increase, resulting in going over budget. Alternatively, the costs may decrease following Father's Day, and the additional funds could be allocated to other terms.

[0021] Advertisers must also keep track of the number of clicks that a listing is getting, e.g., to calculate the conversion rate. If a listing is getting many clicks but few sales, then it could be the case that the listing's description is not sufficiently specific. Alternatively, if a listing is getting too few clicks, it could be the case that other advertisers have entered the marketplace, which has resulted in the listing being at a worse rank than before.

[0022] It is also important for advertisers to keep track of the click through rate (CTR) of listings. For example, a new title or description for a listing may result in a lower CTR if it is less clear than what was there before. Keeping track of the CTR ensures that corrective action can be taken promptly.

[0023] Advertisers must also keep track of their account balance at the pay for performance marketplace. The balance should never reach zero, in order to ensure continued service without interruption. In addition, it is important to keep track of the account balance to ensure that the budget is spent according to plan. For example, if the balance is going down too slowly in the first week, the advertiser can take corrective action to increase the CPC of listings to get back on track.

[0024] There are other marketplace conditions that advertisers must keep track of. These include checking if the bid of a listing is too high for its current rank. For example, an

advertiser A_1 may set the CPC of a listing to \$0.50 for the listing to be at rank 2—advertiser A_2 is at rank 3 with a CPC of \$0.49. A few hours later, A_2 changes the CPC of his listing to \$0.45, while still remaining at rank 3. Advertiser A_1 can now reduce the CPC of his listing from \$0.50 to \$0.46, while still maintaining the listing at rank 2.

[0025] Advertisers must also keep track of the changing costs in the marketplace for different ranks. A rank that was unaffordable earlier may now become affordable, or vice-versa. For example, advertiser A_1 is at rank 5 and wishes to be at rank 3 in order to get higher traffic. The current CPC for rank 3 is \$1.00, and the CPC for rank 4 is \$0.75. A_1 can afford at most \$0.80 for this listing. That is, the advertiser's return on investment (ROI) analysis indicates that anything higher will result in a loss. If the advertiser at rank 3 drops out, A_1 can jump to rank 3 with a CPC of \$0.76, which is within his budget of \$0.80.

[0026] The previous examples illustrate the various actions that advertisers must perform manually to manage their listings. Some advertisers do these tasks several times a day. Some advertisers have a plurality of employees dedicated to the management of their participation in a pay for placement marketplace, monitoring the positions of their listings and adjusting their bids, managing their budget, etc. The manual process of polling of the status of listings, checking the competitors in the marketplace, and checking the account status is time consuming and wasteful. Only some of these concerns need addressing at a given time. Therefore, a need exists for a method and apparatus for advertisers to manage their listings more effectively.

[0027] U.S. application Ser. No. 09/922,028, entitled "System And Method For Providing Place And Price Protection In A Search Result List Generated By A Computer Network Search Engine," filed Aug. 3, 2001, discloses a system which may be referred to as Price and Place Protection. This application is commonly assigned with the present application and is incorporated herein by reference. In the disclosed system, an advertiser's bid does not establish a fixed CPC. Instead, his bid sets the maximum CPC the advertiser will incur. Further, the disclosed embodiments allow the advertiser to specify a desired rank in the search results displayed to the searcher. The rank of a search listing is the ordinal positioning of the search listing among a group of search listings matching the searcher's search term. Higher or better listed search listings are displayed higher on a page and earlier on a number of pages of search listings. The system of the present embodiments determines the actual rankings and actual CPC's. The listings matching a search may then be ranked in descending order of CPC, with priority among listings of equal CPC by chronological seniority.

[0028] If these inefficiencies are not addressed by a marketplace promoter, then an economic incentive remains for advertisers to produce automated services of their own to interact with the account management systems of the marketplace operator to obtain the economic advantage available relative to the limited automated services provided by the marketplace operator. As a further consequence, such a situation provides economic incentive for third parties to produce automated services for advertisers, for a fee, or a cut of the alleged savings produced. This is already happening.

BRIEF SUMMARY

[0029] By way of introduction only, the present embodiments may be referred to collectively as Auto Notification. Auto Notification is an improvement on existing pay for performance marketplace systems. In the basic marketplace system, an advertiser logs on to the advertiser interface and manages his advertising campaign by examining the marketplace information and the information related to his listings. For example, an advertiser can identify a set of terms, their description, and other information, which includes the CPC for each term, which is the amount that the advertiser will pay if a user clicks on the listing. An advertiser can also check the number of clicks at different ranks for a search term, examine the other competitive listings for a term, check his account balance, add funds to his account, etc. Subsequently, when a search term matches a search query received from a searcher, economic value may be given by the advertiser to the marketplace operator.

[0030] The embodiments described herein use the concept of a bid which corresponds to economic value which the advertiser will give when network locations associated with the advertiser is referred to a searcher in response to a query from the searcher. The economic value may be a money amount charged or chargeable to the advertiser, either directly or indirectly. The economic value may be an amount debited from an account of the advertiser. The amount may be a money amount or another value, such as credit points. The economic value may be given by the advertiser to the operator of a database search system or to a third party.

[0031] The economic value is given when one or more network locations, such as advertiser web sites, are referred to a searcher. The referral may be by presenting the network locations on a screen used for data entry and receipt by the searcher, alone or with other search results. This is referred to as an impression. Alternatively, and in an embodiment generally described herein, the referral may occur when the searcher clicks on or clicks through to access the network locations of the advertiser, as will be described in greater detail below. Or the referral may be by some other action taken by the searcher after accessing the network locations of the advertiser.

[0032] The embodiments herein automate many of the steps performed by an advertiser. Currently an advertiser must periodically examine the state of his listings, the state of the marketplace, and his account information, in order to see if any of the conditions that he cares about are true. This manual examination of the marketplace, listings, and his account is time consuming and wasteful, as most of the time no special action is required.

[0033] The disclosed embodiments of Auto Notification enable an advertiser to specify the conditions the advertiser cares about. The system provides an automated agent that acts on behalf of the advertiser, constantly checking if any of the conditions are true. The agent is a software process or application operating in conjunction with data maintained by the marketplace system. If all is well and no conditions are true, then the agent takes no action. Otherwise, the agent makes a note of the condition that is true, and can send a message to alert the advertiser. The message can include means for the advertiser to correct the undesirable conditions, as will be described below. Messages can be sent whenever a condition is true, or they can be aggregated and sent periodically, at the control of the advertiser.

[0034] With Auto Notification, an advertiser need no longer manually search for conditions that are true. Instead, the system automatically notifies the advertiser of the true conditions and possible corrective actions, at the times specified by the advertiser.

[0035] An advertiser can request auto notification for zero or more conditions. Some conditions are related to the listings of the advertiser, and each listing can have zero or more conditions associated with it. In accordance with the present embodiments, each auto notification function has four components:

- [0036]** 1. notification condition: information about the state requiring attention
- [0037]** 2. notification time(s): when the notifications should be sent
- [0038]** 3. notification mode(s): how the advertiser should be notified, and
- [0039]** 4. notification action type(s): the types of corrective actions to include in any notification.

[0040] Notification Condition

[0041] In accordance with the present embodiments, there are nine types of conditions that an advertiser can select from:

- [0042]** 1. position: related to the position of a listing
- [0043]** 2. cost: related to the accumulated costs for some listings
- [0044]** 3. account-balance: related to the funds remaining in advertiser's account (e.g., to pay for listings that are clicked on)
- [0045]** 4. impressions: the number of impressions received by some listings
- [0046]** 5. clicks: the number of clicks received by some listings
- [0047]** 6. CTR: the click through rate of some listings
- [0048]** 7. CPC-too-high: if the cost per click (CPC) of a listing can be reduced without impacting its rank
- [0049]** 8. Average CPC too high: the average CPC, the total cost divided by the total clicks, is higher than some threshold.
- [0050]** 9. rank-CPC: related to the CPC for a given rank and term

[0051] Each condition has its own set of parameters, which are specified by an advertiser. Some of the parameters may have default values, which are at the discretion of the marketplace operator. The parameters for the different conditions are described below.

[0052] A position condition monitors the position of a listing. Each position condition has the following parameters:

- [0053]** 1. listing: the listing whose position is being monitored. This could be a listing of the advertiser, or the listing of some other advertiser.
- [0054]** 2. absolute/relative: an indication of whether the absolute position of the listing is being moni-

tored, or if the position relative to some other listing is being monitored. If the position is relative to another listing, then the other listing is also specified.

[0055] 3. within/without: the condition is true if the listing is within or outside the specified range.

[0056] 4. specific/range: a specific rank or range of ranks. For example, "rank 3" is a specific rank, and "ranks 3 to 5" (inclusive) is a range of ranks, as are "ranks greater than or equal to 4" and "ranks less than 3".

[0057] The following are all examples of position conditions:

[0058] 1. "My listing L_1 is not at rank 3" listing: L_1 , absolute/relative: absolute, within/without: without, specific/range: rank 3.

[0059] 2. "Another listing L_2 is at rank 1" listing: L_2 , absolute/relative: absolute, within/without: within, specific/range: rank 1.

[0060] 3. "My listing L_3 is at ranks 4 through 8 inclusive" listing: L_3 , absolute/relative: absolute, within/without: within, specific/range: ranks 4 through 8.

[0061] 4. "My listing L_4 is more than 2 ranks lower than another listing L_5 " listing: L_4 , absolute/relative: relative to L_5 , within/without: without, specific/range: ranks 1 through 2.

[0062] 5. "My listing L_6 is 3 ranks higher than another listing L_7 " listing: L_6 , absolute/relative: relative to L_7 , within/without: within, specific/range: rank -3 (negative ranks are above the reference rank and positive ranks are below).

[0063] A marketplace operator may provide a variety of user interfaces for entering parameters. For position constraints, a marketplace may provide a simple interface for tracking multiple listings, e.g., to track the change in position of all listings.

[0064] A cost condition monitors the total CPC expenditures for one or more listings of the advertiser in a given time interval. At the start of every time interval the accumulated costs are zero. The starting point of each time interval is at the discretion of the marketplace operator. For example, all hourly intervals could start at the start of every half hour. Each cost condition has the following parameters:

[0065] 1. listings: one or more listings whose CPC expenditure is being monitored.

[0066] 2. limit: the expenditure limit for the accumulated CPCs for all the listings, e.g., \$300.00.

[0067] 3. interval: the time period for the limit, e.g., one week.

[0068] The following are all examples of cost conditions:

[0069] 1. "The CPC charges for listing L_1 exceed \$300.00 in any hour" listings: L_1 , limit: \$300.00, interval: 1 hour

[0070] 2. "The CPC charges for L_2 and L_3 exceed \$195.00 in any month" listings: L_2 and L_3 , limit: \$195.00, interval: 1 month

[0071] The account-balance condition monitors the amount of funds remaining in the account of an advertiser. Some advertisers may be required to pre-pay a deposit, which is used to draw down the CPC charges incurred by the advertiser. An advertiser may periodically replenish his account balance to ensure continual service. Each account-balance condition has the following parameters:

[0072] 1. threshold: the condition is true when the account balance falls below the threshold amount.

[0073] The following are all examples of account-balance conditions:

[0074] 1. "My account balance is less than \$100.00" threshold: \$100.00

[0075] 2. "My account balance is less than \$350.00" threshold: \$350.00

[0076] The impressions condition monitors the aggregate number of impressions for a set of listings of an advertiser in a given interval. At the start of every time interval the accumulated impressions are zero. The starting point of each time interval is at the discretion of the marketplace operator. In one embodiment, an impression is defined as follows. Whenever a user types in a search term, a set of matching search results are presented. The presentation of a listing to a user is counted as an impression. If a listing is on a following page, and the user does not search beyond the current page, then this does not count as an impression. Other definitions may be used as well. If the rank of a listing changes, then the number of impressions for the listing can be reset to zero. This is at the discretion of the advertiser.

[0077] Each impressions condition has the following parameters:

[0078] 1. listings: one or more listings whose aggregate number of impressions is being monitored.

[0079] 2. within/without: whether the condition is true if the number of impressions is within or outside the range.

[0080] 3. range: the range of the impressions being monitored, e.g., 100 to 200.

[0081] 4. interval: the time period for the limit, e.g., 1 day.

[0082] The following are all examples of impressions conditions:

[0083] 1. "Listings L_1 has more than 1000 impressions in one hour"

[0084] listings: L_1 , within/without: without, range: 0 to 1000, interval: 1 hour

[0085] 2. "Listings L_2 , L_3 , and L_4 together have less than 100 impressions in a day"

[0086] listings: L_2 , L_3 , and L_4 , within/without: within, range: 0 to 99, interval: 1 day

[0087] The clicks condition monitors the aggregate number of user clicks for a set of listings of an advertiser in a given interval. At the start of every time interval the accumulated clicks are zero. The starting point of each time interval is at the discretion of the marketplace operator. Whenever a user types in a search term, a set of matching

search results are presented. If a user selects a matching listing by pointing to a hyperlink or typing in a uniform resource locator, this is referred to as clicking on the listing. Other definitions of clicking may be used as well. If a searcher clicks on a matching listing, then this is counted as a click for the listing. If the rank of a listing changes, then the number of clicks for the listing can be reset to zero. This is at the discretion of the advertiser.

[0088] Each clicks condition has the following parameters:

[0089] 1. listings: one or more listings whose number of clicks is being monitored.

[0090] 1. within/without: whether the condition is true if the number of clicks is within or outside the range.

[0091] 2. range: the range of the clicks being monitored, e.g., 1,000 to 4,000.

[0092] 3. interval: the time period for the limit, e.g., 1 quarter.

[0093] The following are all examples of clicks conditions:

[0094] 1. "Listings L_1 has fewer than 100 clicks in one day"

[0095] listings: L_1 , within/without: within, range: 0 to 99, interval: 1 day

[0096] 2. "Listings L_2 and L_3 together have more than 1,500 clicks in a week"

[0097] listings: L_2 , and L_3 , within/without: without, range: 0 to 1,500, interval: 1 week

[0098] The CTR condition monitors the aggregate click through rate for a set of listings of an advertiser over an interval. The aggregate CTR over an interval is the aggregate number of clicks for the interval divided by the aggregate number of impressions for the same interval. When starting to monitor the aggregate CTR, there may be insufficient impressions for valid data. The marketplace operator may select a minimum number of impressions that are required before considering the CTR conditions to be valid.

[0099] Alternatively, an advertiser may specify probability and a margin of error, and from the marketplace operator can calculate the minimum number of clicks required before considering the CTR condition to be valid. For example, the advertiser may specify a 95% probability and a margin of error of 3%. From Statistics we know that if the CTR is a Standard Normal Distribution, there is a 95% probability that a value is between ± 1.96 standard deviations of its mean. So if we take n measurements and get an observed CTR of p' , then

$$1.96 \times \sqrt{\frac{p' \times (1 - p')}{n}} \leq 3\%.$$

[0100] This depends on the observed CTR of p' and can always be achieved by the marketplace operator by waiting for a sufficiently large " n ." Any introductory Statistics text can describe this in detail, for example, "Larsen, Richard J.

and Marx, Morris L. *An Introduction to Mathematical Statistics and Its Applications*, 3rd edition (Jan. 15, 2000) Prentice Hall College Div; ISBN: 0139223037.

[0101] If the rank of a listing changes, then the number of impressions and clicks for the listing can be reset to zero. This is at the discretion of the advertiser.

[0102] Each CTR condition has the following parameters:

[0103] 2. listings: one or more listings whose aggregate CTR is being monitored.

[0104] 3. within/without: whether the condition is true if the aggregate CTR is within or outside the range.

[0105] 4. range: the range of aggregate CTR being monitored, e.g., 1/100 to 1/200.

[0106] 5. interval: The time period for the interval. Data older than the time interval is not considered, e.g., an interval of 1 day would ignore all impressions and clicks older than a day when computing the CTR.

[0107] The following are all examples of CTR conditions:

[0108] 1. "The CTR of listings L_1 is less than 1%, over the last hour"

[0109] listings: L_1 , within/without: within, range: 0 to 1/100, interval: 1 hour.

[0110] 2. "Listings L_2 , and L_3 have an aggregate CTR outside of 1% to 5% over their entire history"

[0111] listings: L_2 , and L_3 , within/without: without, range: 1/100 to 5/100,

[0112] interval: all time.

[0113] 3. "Listing L_4 has a CTR greater than 10% over the last week"

[0114] listing: L_4 , within/without: without, range: 0 to 1/10, interval 1 week.

[0115] A CPC-too-high condition monitors the CPC of one or more listings. The condition is true if the CPC of any monitored listing can be reduced without reducing its rank. For example, if listing L_1 has a CPC of \$1.23 and is at rank 4, and the listing at rank 5 has a CPC of \$1.10, then the CPC of L_1 can be reduced to \$1.11, while still ensuring that L_1 retains rank 4. An advertiser can also specify the size of the gap between the CPC of one its listings and the CPC of the listing below. Each CPC-too-high condition has the following parameters:

[0116] 1. listings: the listings being monitored.

[0117] 2. threshold: the minimum difference between the CPC of a listing and the CPC of the next worse listing.

[0118] The following are all examples of CPC-too-high conditions:

[0119] 1. "Listing L_1 has a CPC higher than \$0.05 compared to the listing below"

[0120] listings: L_1 , threshold: \$0.05

[0121] 2. "Listings L_2 and L_3 have their CPC higher than \$0.01 compared to the listing below"

[0122] listings: L_2 and L_3 , threshold: \$0.01.

[0123] An average CPC-too-high condition monitors the average CPC of one or more listings. The average CPC is the total cost of the listings divided by the total clicks for the listings. The condition is true if the average CPC of all monitored listing is higher than a threshold prescribed by the advertiser. For example, an advertiser can define a condition which is true when the average CPC of all the advertiser's listings is greater than \$1.45. When starting to monitor the average CPC, there may be insufficient impressions and clicks for valid data. The marketplace operator may select a minimum number of impressions and/or clicks that are required before considering the average CPC conditions to be valid. Each average CPC-too-high condition has the following parameters:

[0124] 1. listings: the listings being monitored.

[0125] 2. threshold: the minimum difference between the CPC of a listing and the CPC of the next worse listing.

[0126] 3. interval: the timer period for the limit, e.g., one week.

[0127] The following are all examples of average CPC-too-high conditions:

[0128] 1. "Listings L_1 and L_2 have an average CPC higher than \$0.35 over one day"

[0129] listings: L_1 and L_2 , threshold: \$0.35, interval: 1 day.

[0130] 2. "All my listings have an average CPC higher than \$0.98 over one week"

[0131] listings: all, threshold: \$0.98, interval: 1 week.

[0132] A rank-CPC condition monitors the minimum CPC required to attain a given rank for a search term. The condition is true if a given rank can be achieved with the price threshold specified.

[0133] For example, if listing L_4 is at rank 4 with a CPC of \$1.23 and listing L_5 is at rank 5 with a CPC of \$1.15, then a new listing can be at rank 5 with a CPC of \$1.16. It may be impossible for a new listing to be at a given rank at any price. This can happen, for example, if the CPC of L_4 is the same as the CPC of L_5 . This is because listings are ordered by their CPC, and listings with the same CPC are ordered by their time-stamp (the listing with the earlier time stamp has the better rank). Any new listing will have a time stamp greater than all other listings, and so it cannot have a time stamp in between that of L_4 and L_5 .

[0134] Each rank-CPC condition has the following parameters:

[0135] 1. term: the term being monitored.

[0136] 2. rank: the desired rank

[0137] 3. threshold: the maximum price to be at the rank for term.

[0138] The following are all examples of rank-CPC conditions:

[0139] 1. "rank 3 for the term 'LCD Projector' can be achieved for less than or equal to \$3.50"

[0140] term: LCD Projector, rank: 3, threshold: \$3.50

[0141] 2. "rank 10 for the term 'Garage' can be achieved for less than or equal to \$0.10"

[0142] term: Garage, rank: 10, threshold: \$0.10

[0143] Notification Time(s)

[0144] The previous section described the various notification conditions and the parameters for them. Each Auto Notification specification also includes the notification time(s) for the condition, which is the time(s) at which an advertiser wishes to be notified when the condition is true. Note that the time at which an advertiser is notified is independent of the time at which a condition is true.

[0145] There are two choices when an advertiser can be notified:

[0146] 1. immediately: as soon as a condition becomes true, the advertiser is notified.

[0147] 2. interval: all notifications are aggregated over the specified time interval. The interval includes a period and a time, e.g., hourly at half past the hour, daily at 4:20 p.m., weekly every Friday at 3:45 p.m., etc.

[0148] If no conditions were true during the interval, then the advertiser or marketplace operator can select if no notification should be sent, or if a "no condition true" notification should be sent. Otherwise, all conditions that became true during the interval are recorded, and at the end of the interval the advertiser is notified of these.

[0149] For example, an advertiser may specify that all notifications for a position condition be sent daily. If the position of a monitored listing goes outside the limits specified multiple times during a day, then these are all recorded as they occur, and the advertiser is not sent an immediate notification. At the end of the day these are all gathered and sent to the advertiser.

[0150] Notification Mode

[0151] The previous section defined the notification time, which is the time at which an advertiser is notified of any conditions that may be true. Any such notification is transmitted in one or more possible communication modes. Each Auto Notification specification also includes the notification mode for the condition, which is the communication mode used to notify an advertiser.

[0152] There are five possible modes of communication:

[0153] 1. e-mail: the notice is sent to a set of e-mail addresses prescribed by the advertiser. Each e-mail message can include details of the conditions that are/were true, and links to corrective action that an advertiser can take, e.g., a single click that authenticates the advertisers and automatically makes the corrections.

[0154] 2. instant messaging: the notice is sent to a set of instant-message accounts prescribed by the advertiser. Similar to e-mail, each instant message can include the details of the conditions that are/were true, and links to corrective action that the advertiser can take.

[0155] 3. fax: the notice is faxed to a number prescribed by the advertiser. The fax can include details of the conditions that are/were true, and provide pointers to where the advertiser can go to correct any undesirable conditions, e.g., pointers to the online marketplace system where the advertiser can authenticate himself and then correct any undesirable conditions online.

[0156] 4. page: the notice is paged to a number prescribed by the advertiser. A page is a text or other message sent by radio communication to a portable wireless receiver. The page may be sent through a paging system to a dedicated paging receiver or transceiver, or the page may be sent using a short message service (SMS) operated in conjunction with some cellular radiotelephone systems. The page can provide a brief indication of the conditions that are/were true, and a pointer to the where the advertiser can go to correct any undesirable conditions, e.g., a phone number the advertiser can call.

[0157] 5. phone: the notice is sent to a number prescribed by the advertiser. An automated voice synthesis system can be used to alert the advertiser to the conditions that are/were true. The phone means can offer corrective actions in a menu with touch-tone inputs, e.g., "press 1 to increase your bid to one dollar and thirty two cents to regain position 1, press 2 to . . ." The system may recognize voice inputs directly. The message can also include pointers to where the advertiser can go to correct any undesirable conditions, e.g., pointers to the online marketplace system.

[0158] Notification Action Type

[0159] For each Auto Notification function, an advertiser specifies the condition, notification time, and notification mode. Auto Notification functions also include the notification action type, which is the method that the advertiser can use to correct any undesirable conditions. The actions to correct the condition can be included with the notification, or the notification function can include other instructions to make the corrections. There are six action types:

[0160] 1. active links: these are links that are embedded in the notification, which allow the advertiser to correct the undesirable condition in one click. Preferably, the advertiser is first authenticated before any action is taken. The links can be embedded URLs in an e-mail message, that in one click correct an undesirable condition. For example, a link may be titled "Click here to increase the CPC of the following listing to \$1.43 to restore it to rank 3." The URL of the link points to market operator's system, and includes information about the advertiser and the condition(s) to be corrected. If the advertiser clicks on the link, his identity is verified, and the system performs all the corrective actions automatically without requiring the advertiser to interact with the online marketplace system directly.

[0161] It is applicable to include active links in e-mail notifications and instant messaging notifications.

[0162] 2. inactive links: these are pointers to online locations where an advertiser can go to correct any undesirable conditions. For example, this can be a phone message with a pointer to the URL for the online marketplace system where the advertiser can log in. Once logged in, the advertiser may be presented with a page with active links to correct any undesirable conditions.

[0163] It is applicable to include inactive links in all notification modes.

[0164] 3. e-mail: this is an e-mail template that an advertiser can fill out, indicating what corrective actions (if any) are to be taken, and then e-mail to the address prescribed by the marketplace operator. The template may be included in a notification (e.g., an e-mail notification), or it could be made available through other means, e.g., a web site.

[0165] It is applicable to include e-mail links in all notification modes.

[0166] 4. phone: this is a pointer to a phone number that the advertiser can call to take corrective action. This may be a fully automated system, e.g., with a touch-tone phone and voice recognition, a system with a human operator, or some combination of these.

[0167] It is applicable to include phone links in all notification modes.

[0168] 5. auto-correct: the advertiser is asking the system to automatically take corrective action on his behalf if this condition becomes true. The advertiser also specifies the specifics of the corrective action. This option is only applicable to conditions that can be corrected. For example, an auto-correct action type may instruct the system to add \$500 to the advertiser's account balance, if it gets below the threshold, by automatically charging his credit card.

[0169] 6. relax: the advertiser is asking the system to ignore the current condition, and wants to relax the condition so that this occurrence will not trigger the condition. The marketplace and/or the advertiser can choose how to relax the condition. For example, an advertiser may not care that his listing has fallen to rank 3 from rank 2, but he does want to be notified if it falls further.

[0170] Every notification function can include one or more applicable action types in it. Some action types may not be applicable with some notification modes, e.g., it may not be convenient to include a URL pointer in a phone message. An advertiser may also specify which action types he prefers in a notification.

[0171] The advantage of the Auto Notification system is to implement the following instructions on behalf of participating advertisers:

[0172] 1. Allow me to specify my notification condition(s), notification time(s), notification modes, and notification action types.

[0173] 2. Continually monitor all my conditions to see if any of them are true.

[0174] 3. If any of my conditions become true, make a note of the details of this.

[0175] 4. Send me notifications at the notification time(s) I have specified, if any of my conditions are/were true.

[0176] 5. In each notification include all applicable action types to correct any conditions (that can be corrected). Restrict the action types to those that I have specified I prefer. If I have not given any preferences the marketplace operator may chose to include some or all of the action types with each notification.

[0177] The foregoing discussion of the preferred embodiments has been provided only by way of introduction. Nothing in this section should be taken as a limitation on the following claims, which define the scope of the invention.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[0178] FIG. 1 is a block diagram illustrating the relationship between a large network and one embodiment of the system and method for generating a pay-for-performance search result of the present invention;

[0179] FIG. 2 is a chart of menus, display screens, and input screens used in one embodiment of the present invention;

[0180] FIG. 3 is a flow chart illustrating the advertiser user login process performed in one embodiment of the present invention;

[0181] FIG. 4 is a flow chart illustrating the administrative user login process performed in one embodiment of the present invention;

[0182] FIG. 5 is a diagram of data for an account record for use with one embodiment of the present invention;

[0183] FIG. 6 is a flow chart illustrating a method of adding money to an account record used in one embodiment of the present invention;

[0184] FIG. 7 illustrates an example of a search result list generated by one embodiment of the present invention;

[0185] FIG. 8 is a flow chart illustrating a change bids process used in one embodiment of the present invention;

[0186] FIG. 9 illustrates an example of a screen display used in the change bids process of FIG. 8; and

[0187] FIGS. 10-24 are flow diagrams illustrating operation of a system in accordance with the present embodiments.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

[0188] Methods and systems for generating a pay-for-performance search result determined by a site promoter, such as an advertiser, over a client/server based computer network system are disclosed. The following description is presented to enable any person skilled in the art to make and use the invention. For purposes of explanation, specific

nomenclature is set forth to provide a thorough understanding of the present invention. Descriptions of specific applications are provided only as examples. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[0189] Referring now to the drawings, FIG. 1 is an example of a distributed system 10 configured as client/server architecture used in a preferred embodiment of the present invention. A "client" is a member of a class or group that uses the services of another class or group to which it is not related. In the context of a computer network, such as the Internet, a client is a process (i.e. roughly a program or task) that requests a service which is provided by another process, known as a server program. The client process uses the requested service without having to know any working details about the other server program or the server itself. In networked systems, a client process usually runs on a computer that accesses shared network resources provided by another computer running a corresponding server process. However, it should also be noted that it is possible for the client process and the server process to run on the same computer.

[0190] A "server" is typically a remote computer system that is accessible over a communications medium such as the Internet. The client process may be active in a second computer system, and communicate with the server process over a communications medium that allows multiple clients to take advantage of the information-gathering capabilities of the server. Thus, the server essentially acts as an information provider for a computer network.

[0191] The block diagram of FIG. 1 therefore shows a distributed system 10 comprising a plurality of client computers 12, a plurality of advertiser web servers 14, an account management server 22, and a search engine web server 24, all of which are connected to a network 20. The network 20 will be hereinafter generally referred to as the Internet. Although the system and method of the present invention is specifically useful for the Internet, it should be understood that the client computers 12, advertiser web servers 14, account management server 22, and search engine web server 24 may be connected together through one of a number of different types of networks. Such networks may include local area networks (LANs), other wide area networks (WANs), and regional networks accessed over telephone lines, such as commercial information services. The client and server processes may even comprise different programs executing simultaneously on a single computer.

[0192] The client computers 12 can be conventional personal computers (PCs), workstations, or computer systems of any other size. Each client 12 typically includes one or more processors, memories, input/output devices, and a network interface, such as a conventional modem. The advertiser web servers 14, account management server 22, and the search engine web server 24 can be similarly configured. However, advertiser web servers 14, account

management server **22**, and search engine web server **24** may each include many computers connected by a separate private network. In fact, the network **20** may include hundreds of thousands of individual networks of computers.

[0193] The client computers **12** can execute web browser programs **16**, such as the NAVIGATOR, EXPLORER, or MOSAIC browser programs, to locate the web pages or records **30** stored on advertiser server **14**. The browser programs **16** allow the users to enter addresses of specific web pages **30** to be retrieved. These addresses are referred to as Uniform Resource Locators, or URLs. In addition, once a page has been retrieved, the browser programs **16** can provide access to other pages or records when the user "clicks" on hyperlinks to other web pages. Such hyperlinks are located within the web pages **30** and provide an automated way for the user to enter the URL of another page and to retrieve that page. The pages can be data records including as content plain textual information, or more complex digitally encoded multimedia content, such as software programs, graphics, audio signals, videos, and so forth.

[0194] In a preferred embodiment of the present invention, shown in FIG. 1, client computers **12** communicate through the network **20** with various network information providers, including account management server **22**, search engine server **24**, and advertiser servers **14** using the functionality provided by a HyperText Transfer Protocol (HTTP), although other communications protocols, such as FTP, SNMP, TELNET, and a number of other protocols known in the art, may be used. Preferably, search engine server **24**, account management server **22**, and advertiser servers **14** are located on the World Wide Web.

[0195] As discussed above, at least two types of server are contemplated in a preferred embodiment of the present invention. The first server contemplated is an account management server **22** comprising a computer storage medium **32** and a processing system **34**. A database **38** is stored on the storage medium **32** of the account management server **22**. The database **38** contains advertiser account information. It will be appreciated from the description below that the system and method of the present invention may be implemented in software that is stored as executable instructions on a computer storage medium, such as memories or mass storage devices, on the account management server **22**. Conventional browser programs **16**, running on client computers **12**, may be used to access advertiser account information stored on account management server **22**. Preferably, access to the account management server **22** is accomplished through a firewall, not shown, which protects the account management and search result placement programs and the account information from external tampering. Additional security may be provided via enhancements to the standard communications protocols such as Secure HTTP or the Secure Sockets Layer.

[0196] The second server type contemplated is a search engine web server **24**. A search engine program permits network users, upon navigating to the search engine web server URL or sites on other web servers capable of submitting queries to the search engine web server **24** through their browser program **16**, to type keyword queries to identify pages of interest among the millions of pages available on the World Wide Web. In a preferred embodiment of the present invention, the search engine web server

24 generates a search result list that includes, at least in part, relevant entries obtained from and formatted by the results of the bidding process conducted by the account management server **22**. The search engine web server **24** generates a list of hypertext links to documents that contain information relevant to search terms entered by the user at the client computer **12**. The search engine web server transmits this list, in the form of a web page, to the network user, where it is displayed on the browser **16** running on the client computer **12**. A presently preferred embodiment of the search engine web server may be found by navigating to the web page at URL <http://www.goto.com/>. In addition, the search result list web page, an example of which is presented in FIG. 7, will be discussed below in further detail.

[0197] Search engine web server **24** is connected to the Internet **20**. In a preferred embodiment of the present invention, search engine web server **24** includes a search database **40** comprised of search listing records used to generate search results in response to user queries. In addition, search engine web server **24** may also be connected to the account management server **22**. Account management server **22** may also be connected to the Internet. The search engine web server **24** and the account management server **22** of the present invention address the different information needs of the users located at client computers **12**.

[0198] For example, one class of users located at client computers **12** may be network information providers such as advertising web site promoters or owners having advertiser web pages **30** located on advertiser web servers **14**. These advertising web site promoters, or advertisers, may wish to access account information residing in storage **32** on account management server **22**. An advertising web site promoter may, through the account residing on the account management server **22**, participate in a competitive bidding process with other advertisers. An advertiser may bid on any number of search terms relevant to the content of the advertiser's web site. In one embodiment of the present invention, the relevance of a bidden search term to an advertiser's web site is determined through a manual editorial process prior to insertion of the search listing containing the search term and advertiser web site URL into the database **40**. In an alternate embodiment of the present invention, the relevance of a bidden search term in a search listing to the corresponding web site may be evaluated using a computer program executing at processor **34** of account management server **22**, where the computer program will evaluate the search term and corresponding web site according to a set of predefined editorial rules.

[0199] The higher bids receive more advantageous placement on the search result list page generated by the search engine **24** when a search using the search term bid on by the advertiser is executed. In a preferred embodiment of the present invention, the amount bid by an advertiser comprises a money amount that is deducted from the account of the advertiser for each time the advertiser's web site is accessed via a hyperlink on the search result list page. A searcher "clicks" on the hyperlink with a computer input device to initiate a retrieval request to retrieve the information associated with the advertiser's hyperlink. Preferably, each access or "click" on a search result list hyperlink will be redirected to the search engine web server **24** to associate the "click" with the account identifier for an advertiser. This redirect action, which is not apparent to the searcher, will

access account identification information coded into the search result page before accessing the advertiser's URL using the search result list hyperlink clicked on by the searcher. The account identification information is recorded in the advertiser's account along with information from the retrieval request as a retrieval request event. Since the information obtained through this mechanism conclusively matches an account identifier with a URL in a manner not possible using conventional server system logs known in the art, accurate account debit records will be maintained. Most preferably, the advertiser's web site description and hyperlink on the search result list page is accompanied by an indication that the advertiser's listing is a paid listing. Most preferably, each paid listing displays a "cost to advertiser," which is an amount corresponding to a "price-per-click" paid by the advertiser for each referral to the advertiser's site through the search result list.

[0200] A second class of users at client computers 12 may comprise searchers seeking specific information on the web. The searchers may access, through their browsers 16, a search engine web page 36 residing on web server 24. The search engine web page 36 includes a query box in which a searcher may type a search term comprising one or more keywords. Alternatively, the searcher may query the search engine web server 24 through a query box hyperlinked to the search engine web server 24 and located on a web page stored at a remote web server. When the searcher has finished entering the search term, the searcher may transmit the query to the search engine web server 24 by clicking on a provided hyperlink. The search engine web server 24 will then generate a search result list page and transmit this page to the searcher at the client computer 12.

[0201] The searcher may click on the hypertext links associated with each listing on the search results page to access the corresponding web pages. The hypertext links may access web pages anywhere on the Internet, and include paid listings to advertiser web pages 18 located on advertiser web servers 14. In a preferred embodiment of the present invention, the search result list also includes non-paid listings that are not placed as a result of advertiser bids and are generated by a conventional World Wide Web search engine, such as the INKTOMI, LYCOS, or YAHOO! search engines. The non-paid hypertext links may also include links manually indexed into the database 40 by an editorial team. Most preferably, the non-paid listings follow the paid advertiser listings on the search results page.

[0202] FIG. 2 is a diagram showing menus, display screens, and input screens presented to an advertiser accessing the account management server 22 through a conventional browser program 16. The advertiser, upon entering the URL of the account management server 22 into the browser program 16 of FIG. 1, invokes a login application, discussed below as shown at screen 110 of FIG. 2, running on the processing system 34 of the server 22. Once the advertiser is logged-in, the processing system 34 provides a menu 120 that has a number of options and further services for advertisers. These items, which will be discussed in more detail below, cause routines to be invoked to either implement the advertiser's request or request further information prior to implementing the advertiser's request. In one embodiment of the present invention, the advertiser may access several options through menu 120, including requesting customer service 130, viewing advertiser policies 140,

performing account administration tasks 150, adding money to the advertiser's account 160, managing the account's advertising presence on the search engine 170, and viewing activity reports 180. Context-specific help 190 may also generally be available at menu 120 and all of the above-mentioned options.

[0203] The login procedure of the preferred embodiment of the present invention is shown in FIGS. 3 and 4 for two types of user. FIG. 3 shows the login procedures 270 for an advertiser. FIG. 4 shows the login procedures 290 for an administrator managing and maintaining the system and method of the present invention. As discussed above, the advertiser or administrator at a client computer 12 must first use a browser program at steps 271 or 291 to access the account management server. After the advertiser navigates to the URL of the login page to start the login process at step 272 or 292, the processing system 34 of the account management server 22 invokes a login application at steps 274 or 294. According to this application, the processor provides an input screen 110 (FIG. 2) that requests the advertiser's or administrator's user name and password. These items of information are provided at steps 276 or 296 to a security application known in the art for the purpose of authentication, based on the account information stored in a database stored in storage 32 of account management server 22.

[0204] According to FIG. 3, after the user has been authenticated as an advertiser, the advertiser is provided with the menu screen 120 of FIG. 2 and limited read/write access privileges only to the corresponding advertiser account, as shown in step 278. The advertiser login event 278 may also be recorded in step 280 in an audit trail data structure as part of the advertiser's account record in the database. The audit trail is preferably implemented as a series of entries in database 38, where each entry corresponds to an event wherein the advertiser's account record is accessed. Preferably, the audit trail information for an account record may be viewed by the account owner and other appropriate administrators.

[0205] However, if the user is authenticated as an administrator in step 295 of FIG. 4, the administrator is provided with specified administrative access privileges to all advertiser accounts as shown in step 296. The administrator login event 296 is recorded in step 297 in the audit trail data structure portion of the administrator's account record. This audit trail is preferably implemented as a series of entries in database 38, where each entry corresponds to an event wherein the administrator's account record is accessed. Most preferably, the administrator's audit trail information may be viewed by the account owner and other appropriate administrators.

[0206] Furthermore, instead of the general advertiser main menu shown to the authenticated advertiser users in step 282, the authenticated administrator is provided in step 298 with access to search the database 38 of advertiser accounts. Preferably, a database search interface is provided to the administrator that enables the administrator to select an advertiser account to monitor. For example, the interface may include query boxes in which the administrator may enter an account number or username or contact name corresponding to an account the administrator wishes to access. When the administrator selects an advertiser account

to monitor in step 299, the administrator is then brought to the main advertiser page 120 of FIG. 2, which is also seen by the advertisers.

[0207] Access to the account information 32 located on the account management server 22 is restricted to users having an account record on the system, as only those users are provided with a valid login name and password. Password and login name information is stored along with the user's other account information in the database 38 of the account management server 22, as shown in FIG. 1. Account information, including a login user name and password, is entered in the database 38 of FIG. 1 via a separate online registration process that is outside the scope of the present invention.

[0208] FIG. 5 is a diagram showing the types of information contained in each advertiser account record 300 in the database. First, an advertiser account record 300 contains a username 302 and a password 304, used for online authentication as described above. The account record also contains contact information 310 (e.g., contact name, company name, street address, phone, e-mail address).

[0209] Contact information 310 is preferably utilized to direct communications to the advertiser when the advertiser has requested notification of key advertiser events under the notification option, discussed below. The account record 300 also contains billing information 320 (e.g., current balance, credit card information). The billing information 320 contains data accessed when the advertiser selects the option to add money to the advertiser's account. In addition, certain billing information, such as the current balance, may trigger events requiring notification under the notification option. The audit trail section 325 of an account record 300 contains a list of all events wherein the account record 300 is accessed. Each time an account record 300 is accessed or modified, by an administrator or advertiser a short entry describing the account access and/or modification event will be appended to the audit trail section 330 of the administrator or advertiser account that initiated the event. The audit trail information may then be used to help generate a history of transactions made by the account owner under the account.

[0210] The advertising information section 330 contains information needed to conduct the online bidding process of the present invention, wherein a position is determined for a web site description and hyperlink within a search result list generated by a search engine. The advertising data 330 for each user account 300 may be organized as zero or more subaccounts 340. Each subaccount 340 comprises at least one search listing 344. Each search listing corresponds to a bid on a search term. An advertiser may utilize subaccounts to organize multiple bids on multiple search terms, or to organize bids for multiple web sites. Subaccounts are also particularly useful for advertisers seeking to track the performance of targeted market segments. The subaccount superstructure is introduced for the benefit of the advertisers seeking to organize their advertising efforts, and does not affect the method of operation of the present invention. Alternatively, the advertising information section need not include the added organizational layer of subaccounts, but may simply comprise one or more search listings.

[0211] The search listing 344 corresponds to a search term/bid pairing and contains key information to conduct the

online competitive bidding process. Preferably, each search listing comprises the following information: search term 352, web site description 354, URL 356, bid amount 358, and a title 360. The search term 352 comprises one or more keywords which may be common words in English (or any other language). Each keyword in turn comprises a character string. The search term is the object of the competitive online bidding process. The advertiser selects a search term to bid on that is relevant to the content of the advertiser's web site. Ideally, the advertiser may select a search term that is targeted to terms likely to be entered by searchers seeking the information on the advertiser's web site, although less common search terms may also be selected to ensure comprehensive coverage of relevant search terms for bidding.

[0212] The web site description 354 is a short textual description (preferably less than 190 characters) of the content of the advertiser's web site and may be displayed as part of the advertiser's entry in a search result list. The search listing 344 may also contain a title 360 of the web site that may be displayed as the hyperlinked heading to the advertiser's entry in a search result list. The URL 356 contains the Uniform Resource Locator address of the advertiser's web site. When the user clicks on the hyperlink provided in the advertiser's search result list entry, the URL is provided to the browser program. The browser program, in turn, accesses the advertiser's web site through the redirection mechanism discussed above. The URL may also be displayed as part of the advertiser's entry in a search result list.

[0213] The bid amount 358 preferably is a money amount bid by an advertiser for a listing. This money amount is deducted from the advertiser's prepaid account or is recorded for advertiser accounts that are invoiced for each time a search is executed by a user on the corresponding search term and the search result list hyperlink is used to refer the searcher to the advertiser's web site. Finally, a rank value is a value generated dynamically, preferably by the processing system 34 of the account management server 22 shown in FIG. 1, each time an advertiser places a bid or a search enters a search query. The rank value of an advertiser's search listing determines the placement location of the advertiser's entry in the search result list generated when a search is executed on the corresponding search term. Preferably, rank value is an ordinal value determined in a direct relationship to the bid amount 358; the higher the bid amount, the higher the rank value, and the more advantageous the placement location on the search result list. Most preferably, the rank value of 1 is assigned to the highest bid amount with successively higher ordinal values (e.g., 2, 3, 4, . . .) associated with successively lower ranks and assigned to successively lower bid amounts.

[0214] Once logged in, an advertiser can perform a number of straightforward tasks set forth in menu 120 of FIG. 2, including viewing a list of rules and policies for advertisers, and requesting customer service assistance. These items cause routines to be invoked to implement the request. For example, when "Customer Service" is selected, an input screen 130 is displayed to allow the advertiser to select the type of customer service requested. In addition, forms may be provided on screen 130 so that an advertiser may type a customer comment into a web-based input form.

[0215] When "View Advertiser Policies" is selected, a routine will be invoked by processing system 34 of the

account management server 22FIG. 1. As shown in FIG. 2, the routine will display an informational web page 140. The web page 140 sets forth the advertiser policies currently in effect (e.g., "All search listing descriptions must clearly relate to the search term").

[0216] Menu 120 of FIG. 2 also includes an "Account Administration" selection 150 which allows an advertiser, among other things, to view and change the advertiser's contact information and billing information, or update the advertiser's access profile, if any. Web-based forms well known in the art and similar to those discussed above are provided for updating account information.

[0217] The "Account Administration" menu also includes a selection enabling an advertiser to view the transaction history of the advertiser's account. Under the "View Transaction History" selection, the advertiser may invoke routines to view a listing of past account transactions (e.g., adding money to account, adding or deleting bid search terms, or changing a bid amount). Additional routines may be implemented to permit advertisers to display a history of transactions of a specified type, or that occur within a specified time. The transaction information may be obtained from the audit trail list 325 of FIG. 5, described above. Clickable buttons that may be implemented in software, web-based forms, and/or menus may be provided as known in the art to enable advertisers to specify such limitations.

[0218] In addition, the "Account Administration" menu 150 of FIG. 2 includes a selection enabling an advertiser to set notification options. Under this selection, the advertiser may select options that will cause the system to notify the advertiser when certain key events have occurred. For example, the advertiser may elect to set an option to have the system send conventional electronic mail messages to the advertiser when the advertiser's account balance has fallen below a specified level. In this manner, the advertiser may receive a "warning" to replenish the account before the account is suspended (meaning the advertiser's listings will no longer appear in search result lists). Another key event for which the advertiser may wish notification is a change in position of an advertiser's listing in the search result list generated for a particular search term. For example, an advertiser may wish to have the system send a conventional electronic mail message to the advertiser if the advertiser has been outbid by another advertiser for a particular search term (meaning that the advertiser's listing will appear in a position farther down on the search result list page than previously). When one of the system-specified key events occurs, a database search is triggered for each affected search listing. The system will then execute the appropriate notification routine in accordance with the notification options specified in the advertiser's account.

[0219] Referring back to FIG. 2, a selection also appears in menu 120 that permits an advertiser to add money to the advertiser's account, so that the advertiser will have funds in their account to pay for referrals to the advertiser's site through the search results page. Preferably, only advertisers with funds in their advertiser's accounts may have their paid listings included in any search result lists generated. Most preferably, advertisers meeting selected business criteria may elect, in place of maintaining a positive account balance at all times, incur account charges regardless of account balance and pay an invoiced amount at regular intervals

which reflects the charges incurred by actual referrals to the advertiser's site generated by the search engine. The process that is executed when the "Add Money to Account" selection is invoked is shown in further detail in FIG. 6, beginning at step 602. When the "Add Money to Account" selection is clicked in step 604, a function is invoked which receives data identifying the advertiser and retrieves the advertiser's account from the database. The executing process then stores the advertiser's default billing information and displays the default billing information for the advertiser in step 606. The displayed billing information includes a default amount of money to be added, a default payment type, and default instrument information.

[0220] In the preferred embodiment of the present invention, an advertiser may add funds online and substantially in real time through the use of a credit card, although the use of other payment types are certainly well within the scope of the present invention. For example, in an alternate embodiment of the present invention, advertisers may add funds to their account by transferring the desired amount from the advertiser's bank account through an electronic funds verification mechanism known in the art such as debit cards, in a manner similar to that set forth in U.S. Pat. No. 5,724,424 to Gifford. In another alternate embodiment of the present invention, advertisers can add funds to their account using conventional paper-based checks. In that case, the additional funds may be updated in the account record database through manual entry. The instrument information includes further details regarding the type of payment. For example, for a credit card, the instrument information may include data on the name of the credit card (e.g., MasterCard, Visa, or American Express), the credit card number, the expiration date of the credit card, and billing information for the credit card (e.g., billing name and address). In a preferred embodiment of the present invention, only a partial credit card number is displayed to the advertiser for security purposes.

[0221] The default values displayed to the advertiser are obtained from a persistent state, e.g., stored in the account database. In an embodiment of the present invention, the stored billing information values may comprise the values set by the advertiser the last (e.g. most recent) time the process of adding money was invoked and completed for the advertiser's account. The default billing information is displayed to the advertiser in a web-based form. The advertiser may click on the appropriate text entry boxes on the web-based form and make changes to the default billing information. After the advertiser completes the changes, the advertiser may click on a hyperlinked "Submit" button provided on the form to request that the system update the billing information and current balance in step 608. Once the advertiser has requested an update, a function is invoked by the system which validates the billing information provided by the advertiser and displays it back to the advertiser for confirmation, as shown in step 610. The confirmation billing information is displayed in read-only form and may not be changed by the advertiser.

[0222] The validation step functions as follows. If payment is to be debited from an advertiser's external account, payment may be authenticated, authorized and completed using the system set forth in U.S. Pat. No. 5,724,424 to Gifford. However, if the payment type is by credit card, a validating algorithm is invoked by the system, which validates the credit card number using a method such as that set

forth in U.S. Pat. No. 5,836,241 to Stein et al. The validating algorithm also validates the expiration date via a straightforward comparison with the current system date and time. In addition, the function stores the new values in a temporary instance prior to confirmation by the advertiser.

[0223] Once the advertiser ascertains that the displayed data is correct, the advertiser may click on a "Confirm" button provided on the page to indicate that the account should be updated in step 612. In step 612, a function is invoked by the system which adds money to the appropriate account balance, updates the advertiser's billing information, and appends the billing information to the advertiser's payment history. The advertiser's updated billing information is stored to the persistent state (e.g., the account record database) from the temporary instance.

[0224] Within the function invoked at step 612, a credit card payment function may be invoked by the system at step 614. In an alternate embodiment of the present invention, other payment functions such as debit card payments may be invoked by defining multiple payment types depending on the updated value of the payment type.

[0225] If the payment type is credit card, the user's account is credited immediately at step 616, the user's credit card having already been validated in step 610. A screen showing the status of the add money transaction is displayed, showing a transaction number and a new current balance, reflecting the amount added by the just completed credit card transaction.

[0226] In an alternate embodiment of the present invention, after the money has been added to the account, the amount of money added to the account may be allocated between subaccounts the end of the add money process at step 616. If the advertiser has no subaccounts, all of the money in the account is a general allocation. However, if the advertiser has more than one subaccount, the system will display a confirmation and default message prompting the advertiser to "Allocate Money Between Subaccounts".

[0227] The menu selection "Allocate Money Between Subaccounts" may be invoked when money is added to the advertiser account after step 616 of FIG. 6, or it may be invoked within the "Account Management" menu 170 shown in FIG. 2. The "Account Management" menu 170 is accessible from the Advertiser Main Page 120, as shown in FIG. 2. This "Allocate Money Between Subaccounts" menu selection permits an advertiser to allocate current and any pending balances of the advertiser's account among the advertiser's subaccounts. The system will then update the subaccount balances. The current balance allocations will be made in real time, while the pending balance allocations will be stored in the persistent state. A routine will be invoked to update the subaccount balances to reflect the pending balance allocations when the payment for the pending balance is processed. Automatic notification may be sent to the advertiser at that time, if requested. This intuitive online account management and allocation permits advertisers to manage their online advertising budget quickly and efficiently. Advertisers may replenish their accounts with funds and allocate their budgets, all in one easy web-based session. The computer-based implementation eliminates time consuming, high cost manual entry of the advertiser's account transactions.

[0228] The "Allocate Money Between Subaccounts" routine begins when an advertiser indicates the intent to allocate money by invoking the appropriate menu selection at the execution points indicated above. When the advertiser indicates the intent to allocate, a function is invoked by the system to determine whether there are funds pending in the current balance (i.e., unactivated account credits) that have not yet been allocated to the advertiser's subaccounts, and displays the balance selection options. In a preferred embodiment of the present invention, an account instance is created and a pending current balance account field is set from the persistent state.

[0229] If there are no unallocated pending funds, the system may display the current available balances for the account as a whole as well as for each subaccount. The advertiser then distributes the current available balance between subaccounts and submits a request to update the balances. A function is invoked which calculates and displays the current running total for subaccount balances. The current running total is stored in a temporary variable which is set to the sum of current balances for all subaccounts for the specified advertiser. The function also validates the new available subaccount balances to make sure that the total does not exceed the authorized amount. If the new advertiser-set available subaccount balances does not exceed the authorized amount, a function is invoked which will update all of the subaccount balances in the persistent state and display the update in read-only format.

[0230] If there are pending funds in the current account balance, the pending funds must be allocated separately from the available current balance. The pending funds will then be added into the available current balance when the funds are received. The function must therefore prompt the advertiser to choose between allocating pending funds or allocating available funds. The allocating pending funds selection works in much the same manner as the allocating available funds selection outlined above. After the advertiser chooses to allocate pending funds, a routine is invoked to display current pending balances for the account and the subaccounts. The advertiser distributes the pending subaccount balances between campaigns and submits a request to update the balances. A function is invoked which calculates and displays the current running totals for the pending subaccount balances. This function also validates the new pending subaccount allocations to make sure that the allocations do not exceed any authorized amount. The current running total of pending allocations is set to the sum of current pending balances for all subaccounts for the advertiser. If the new user-set pending subaccount balances or the total of such balances do not exceed any authorized amount, the function will update all of the pending subaccount allocations in the persistent state, e.g. the advertiser's account in the database, and display the update in read-only format.

[0231] As indicated above and shown in FIG. 2, a routine displaying the account management menu 170 may be invoked from the advertiser main menu 120. Aside from the "Allocate Money Between Subaccounts" selection described above, the remaining selections all use to some extent the search listings present in the advertiser's account on the database, and may also affect the advertiser's entry in the search result list. Thus, a further description of the search result list generated by the search engine is needed at this point.

[0232] When a remote searcher accesses the search query page on the search engine web server **24** and executes a search request according to the procedure described previously, the search engine web server **24** preferably generates and displays a search result list where the “canonicalized” entry in search term field of each search listing in the search result list exactly matches the canonicalized search term query entered by the remote searcher. The canonicalization of search terms used in queries and search listings removes common irregularities of search terms entered by searchers and web site promoters, such as capital letters and pluralizations, in order to generate relevant results. However, alternate schemes for determining a match between the search term field of the search listing and the search term query entered by the remote searcher are well within the scope of the present invention. For example, string matching algorithms known in the art may be employed to generate matches where the keywords of the search listing search term and the search term query have the same root but are not exactly the same (e.g., computing vs. computer). Alternatively a thesaurus database of synonyms may be stored at search engine web server **24**, so that matches may be generated for a search term having synonyms. Localization methodologies may also be employed to refine certain searches. For example, a search for “bakery” or “grocery store” may be limited to those advertisers within a selected city, zip code, or telephone area code. This information may be obtained through a cross-reference of the advertiser account database stored at storage **32** on account management server **22**. Finally, internationalization methodologies may be employed to refine searches for users outside the United States. For example, country or language-specific search results may be generated, by a cross-reference of the advertiser account database, for example.

[0233] An example of a search result list display used in an embodiment of the present invention is shown in **FIG. 7**, which is a display of the first several entries resulting from a search for the term “zip drives”. As shown in **FIG. 7**, a single entry, such as entry **710a** in a search result list consists of a description **720** of the web site, preferably comprising a title and a short textual description, and a hyperlink **730** which, when clicked by a searcher, directs the searcher’s browser to the URL where the described web site is located. The URL **740** may also be displayed in the search result list entry **710a**, as shown in **FIG. 7**. The “click through” of a search result item occurs when the remote searcher viewing the search result item display **710** of **FIG. 7** selects, or “clicks” on the hyperlink **730** of the search result item display **710**. In order for a “click through” to be completed, the searcher’s click should be recorded at the account management server and redirected to the advertiser’s URL via the redirect mechanism discussed above.

[0234] Search result list entries **710a-710h** may also show the rank value of the advertiser’s search listing. The rank value is an ordinal value, preferably a number, generated and assigned to the search listing by the processing system **34** of **FIG. 1**. Preferably, the rank value is assigned through a process, implemented in software, that establishes an association between the bid amount, the rank, and the search term of a search listing. The process gathers all search listings that match a particular search term, sorts the search listings in order from highest to lowest bid amount, and assigns a rank value to each search listing in order. The highest bid amount receives the highest rank value, the next

highest bid amount receives the next highest rank value, proceeding to the lowest bid amount, which receives the lowest rank value. Most preferably, the highest rank value is 1 with successively increasing ordinal values (e.g., 2, 3, 4, . . .) assigned in order of successively decreasing rank. The correlation between rank value and bid amount is illustrated in **FIG. 7**, where each of the paid search list entries **710a** through **710f** display the advertiser’s bid amount **750a** through **750f** for that entry. Preferably, if two search listings having the same search term also have the same bid amount, the bid that was received earlier in time will be assigned the higher rank value. Unpaid listings **710g** and **710h** do not display a bid amount and are displayed following the lowest-ranked paid listing. Preferably, unpaid listings are displayed if there are an insufficient number of listings to fill the 40 slots in a search results page. Unpaid listings are generated by a search engine utilizing objective distributed database and text searching algorithms known in the art. An example of such a search engine may be operated by Inktomi Corporation. The original search query entered by the remote searcher is used to generate unpaid listings through the conventional search engine.

[0235] As shown in the campaign management menu **170** of **FIG. 2**, several choices are presented to the advertiser to manage search listings. First, in the “Change Bids” selection, the advertiser may change the bid of search listings currently in the account. The process invoked by the system for the change bids function is shown in **FIG. 8**. After the advertiser indicates the intent to change bids by selecting the “Change Bids” menu option, the system searches the user’s account in the database and displays the search listings for the entire account or a default subaccount in the advertiser’s account, as shown in step **810**. Search listings may be grouped into subaccounts defined by the advertiser and may comprise one or more search listings. Only one subaccount may be displayed at a time. The display should also preferably permit the advertiser to change the subaccount selected, as shown in step **815**. The screen display will then show the search listings for the selected subaccount, as indicated in step **820**.

[0236] An example of screen display shown to the advertiser in step **810** is shown in **FIG. 9** and will be discussed below. To change bids, the advertiser user may specify new bids for search terms for which the advertiser already has an existing bid by entering a new bid amount into the new bid input field for the search term. The advertiser-entered bid changes are displayed to the advertiser at step **820** of **FIG. 8** as discussed above. To update the bids for the display page, the advertiser requests, at step **830** of **FIG. 8**, to update the result of changes. The advertiser may transmit such a request to the account management server by a variety of means, including clicking on a button graphic.

[0237] As shown in step **840** of **FIG. 8**, upon receiving the request to update the advertiser’s bids, the system calculates the new current bid amounts for every search listing displayed, the rank values, and the bid amount needed to become the highest ranked search listing matching the search term field. Preferably, the system then presents a display of changes at step **850**. After the user confirms the changes, the system updates the persistent state by writing the changes to the account in the database.

[0238] The search listing data is displayed in tabular format, with each search listing corresponding to one row of

the table **900**. The search term **902** is displayed in the leftmost column, followed by the current bid amount **904**, and the current rank **906** of the search listing. The current rank is followed by a column entitled "Bid to become #1"**907**, defined as the bid amount needed to become the highest ranked search listing for the displayed search term. The rightmost column of each row comprises a new bid input field **908** which is set initially to the current bid amount.

[0239] As shown in **FIG. 9**, the search listings may be displayed as "subaccounts." Each subaccount comprises one search listing group, with multiple subaccounts residing within one advertiser account. Each subaccount may be displayed on a separate display page having a separate page. The advertiser should preferably be able to change the subaccount being displayed by manipulating a pull-down menu **910** on the display shown in **FIG. 9**. In addition, search listing groups that cannot be displayed completely in one page may be separated into pages which may be individually viewed by manipulating pull-down menu **920**. Again, the advertiser should preferably be able to change the page displayed by clicking directly on a pull-down menu **920** located on the display page of **FIG. 9**. The advertiser may specify a new bid for a displayed search listing by entering a new bid amount into the new bid input field **908** for the search listing. To update the result of the advertiser-entered changes, the advertiser clicks on button graphic **912** to transmit an update request to the account management server, which updates the bids as described above.

[0240] Many of the other selections listed in the "Account Management" menu **170** of **FIG. 2** function as variants of the "Change Bid" function described above. For example, if the advertiser selects the "Change Rank Position" option, the advertiser may be presented with a display similar to the display of **FIG. 9** used in the "Change Bid" function. However, in the "Change Rank Position" option, the "New Bid" field would be replaced by a "New Rank" field, in which the advertiser enters the new desired rank position for a search term. After the advertiser requests that the ranks be updated, the system then calculates a new bid price by any of a variety of algorithms easily available to one skilled in the art. For example, the system may invoke a routine to locate the search listing in the search database having the desired rank/search term combination, retrieve the associated bid amount of said combination, and then calculate a bid amount that is N cents higher, where N=1, for example. After the system calculates the new bid price and presents a read-only confirmation display to the advertiser, the system updates the bid prices and rank values upon receiving approval from the advertiser.

[0241] The "Modify Listing Component" selection on Account Management menu **170** of **FIG. 2** may also generate a display similar to the format of **FIG. 9**. When the advertiser selects the "Modify Listing Component" option, the advertiser may input changes to the URL, title, or description of a search listing via web-based forms set up for each search listing. Similar to the process discussed above, the forms for the URL, title, and description fields may initially contain the old URL, title and description as default values. After the advertiser enters the desired changes, the advertiser may transmit a request to the system to update the changes. The system then displays a read-only confirmation

screen, and then writes the changes to the persistent state (e.g., the user account database) after the advertiser approves the changes.

[0242] A process similar to those discussed above may be implemented for changing any other peripheral options related to a search listing; for example, changing the matching options related to a bid search term. Any recalculations of bids or ranks required by the changes may also be determined in a manner similar to the processes discussed above.

[0243] In the "Delete Bidded Search Term" option, the system retrieves all of the search listings in the account of the advertiser and displays the search listings in an organization and a format similar to the display of **FIG. 9**. Each search listing entry may include, instead of the new bid field, a check box for the advertiser to click on. The advertiser would then click to place a check (X) mark next to each search term to be deleted, although any other means known in the art for selecting one or more items from a list on a web page may be used. After the advertiser selects all the search listings to be deleted and requests that the system update the changes, the system preferably presents a read-only confirmation of the requested changes, and updates the advertiser's account only after the advertiser approves the changes. The "deleted" search listings are removed from the search database **36** and will not appear in subsequent searches. However, the search listing will remain as part of the advertiser's account record for billing and account activity monitoring purposes.

[0244] In the "Add Bidded Search Term" option, the system provides the advertiser with a display having a number of entry fields corresponding to the elements of a search listing. The advertiser then enters into each field information corresponding to the respective search listing element, including the search term, the web site URL, the web site title, the web site description, and the bid amount, as well as any other relevant information. After the advertiser has completed entering the data and has indicated thus to the system, the system returns a read-only confirmation screen to the advertiser. The system then creates a new search listing instance and writes it into the account database and the search database upon receiving approval from the advertiser.

[0245] Preferably, the "Account Management" menu **170** of **FIG. 2** provides a selection for the advertiser to "Get Suggestions On Bidded Search Term". In this case, the advertiser enters a bidded search term into a form-driven query box displayed to the advertiser. The system reads the search term entered by the advertiser and generates a list of additional related search terms to assist the advertiser in locating search terms relevant to the content of the advertiser's web site. Preferably, the additional search terms are generated using methods such as a string matching algorithm applied to a database of bidded search terms and/or a thesaurus database implemented in software. The advertiser may select search terms to bid on from the list generated by the system. In that case, the system displays to the advertisers the entry fields described above for the "Add Bidded Search Term" selection, with a form for entering a search listing for each search term selected. Preferably, the selected search term is inserted as a default value into the form for each search listing. Default values for the other search listing components may also be inserted into the forms if desired.

[0246] The “Account Management” menu 170 of FIG. 2 also preferably provides advertisers with a “Project Expenses” selection. In this selection, the advertiser specifies a search listing or subaccount for which the advertiser would like to predict a “daily run rate” and “days remaining to expiration.” The system calculates the projections based on a cost projection algorithm, and displays the predictions to the advertiser on a read-only screen. The predictions may be calculated using a number of different algorithms known in the art. However, since the cost of a search listing is calculated by multiplying the bid amount by the total number of clicks received by the search listing at that bid amount during a specified time period, every cost projection algorithm must generally determine an estimated number of clicks per month (or other specified time period) for a search listing. The clicks on a search listing may be tracked via implementation of a software counting mechanism as is well known in the art. Clicks for all search listings may be tracked over time, this data may be used to generate estimated numbers of clicks per month overall, and for individual search terms. For a particular search term, an estimated number of searches per day is determined and is multiplied by the cost of a click. This product is then multiplied by a ratio of the average number of clicks over the average number of impressions for the rank of the search listing in question to obtain a daily run rate. The current balance may be divided by the daily run rate to obtain a projected number of days to exhaustion or “expiration” of account funds.

[0247] One embodiment of the present invention bases the cost projection algorithm on a simple predictor model that assumes that every search term performs in a similar fashion. This model assumes that the rank of the advertiser’s search listing will remain constant and not fluctuate throughout the month. This algorithm has the advantages of being simple to implement and fast to calculate. The predictor model is based on the fact that the click through rate, e.g. the total number of clicks, or referrals, for a particular searcher listing, is considered to be a function of the rank of the search listing. The model therefore assumes that the usage curve of each search term, that is, the curve that result when the number of clicks on a search listing is plotted against the rank of the search listing, is similar to the usage curve for all search terms. Thus, known values extrapolated over time for the sum of all clicks for all search terms, the sum of all clicks at a given rank for all search terms, and the sum of all clicks for the selected search term may be employed in a simple proportion to determine the total of all clicks for the given rank for the selected search term. The estimated daily total of all clicks for the selected search term at the selected rank is then multiplied by the advertiser’s current bid amount for the search term at that rank to determine a daily expense projection. In addition, if particular search terms or classes of search terms are known to differ markedly from the general pattern, correction values specific to the search term, advertiser, or other parameter may be introduced to fine-tune the projected cost estimate.

[0248] Finally, the “Account Management” menu 170 of FIG. 2 provides several selections to view information related to the advertiser’s campaigns. The “View Subaccount Information” selection displays read-only information related to the selected subaccount. The “View Search Term List” selection displays the list of the advertiser’s selected search terms along with the corresponding URLs, bid price, and rank, with the search terms preferably grouped by

subaccount. The advertiser may also view current top bids for a set of search terms selected from a list of search terms from a read-only display generated by the system upon receiving the requested search terms from the advertiser.

[0249] For an advertiser who requires a more comprehensive report of search listing activity, the “View Report” option may be selected from the Advertiser Main Page 120 of FIG. 2. In an embodiment of the present invention, the “View Report” options generate reports comprehensive for up to one year preceding the current date. For example, daily reports are available for the each of the immediately preceding 7 days, weekly reports for the preceding four weeks, monthly reports for the preceding twelve months, and quarterly reports for the last four quarters. Additional reports may also be made available depending on advertiser interest. Other predefined report types may include activity tracked during the following time periods: Since Inception of the Account, Year To Date, Yearly, Quarter To Date, Month To Date, and Week to Date. Report Categories may include a Detail Report, viewable by Advertiser Account, by Search Listing, and by URL, and a Summary Report, viewable by Advertiser Account and by Subaccount. The reports may include identification data such as advertiser account and subaccount name, the dates covered by the report and the type of report. In addition, the reports may include key search listing account data such as current balance, pending current balance, average daily account debit, and run rate. Furthermore, the reports may also include key data, such as: search terms, URLs, bids, current ranks, and number of clicks, number of searches done for the search term, number of impressions (times that the search listing appeared in a search result list), and click through rate (defined as Number of Clicks/Number of Impressions). Preferably, the report is available in at least HTML view options for viewing via a browser program, printing, or downloading. Note, however, that other view options may be made available, such as Adobe Acrobat, PostScript, ASCII text, spreadsheet interchange formats (e.g., CSV, tab-delimited), and other well-known formats.

[0250] When the advertiser has selected the “View Report” option, the system invokes a function which displays a list of available report types, dates, categories, and view options. The system preferably creates a report instance with the following fields, all of which are initially set to null: report type, report date, report category, and view option. Once the advertiser has defined the parameters described above, the system invokes a function to generate the requested report, based on the advertiser-set parameters, and to display the report, based on the view option parameter.

[0251] Finally, a preferred embodiment of the present invention implements an option for context specific help that the advertiser may request at any time the advertiser is logged in. The help option may be implemented as a small icon or button located on the system generated display page. The advertiser may click on the icon or button graphic on the display page to request help, upon which the system generates and displays a help page keyed to the function of the particular display the user is viewing. The help may be implemented as separate display pages, a searchable index, dialog boxes, or by any other methods well known in the art.

[0252] FIGS. 10-24 are flow diagrams illustrating procedures which may be used to implement an automatic noti-

fication functionality to the system described above. In accordance with the automatic notification functionality, an advertiser who has one or more associated search listings stored in a search listings database may specify one or more conditions related to the one or more search listings. When a condition becomes true or the automatic notification functionality is otherwise actuated, a notification is sent to the advertiser. The advertiser may respond in any appropriate manner or not respond at all. The automatic notification is an independent feature under the advertiser's control which operates automatically on behalf of the advertiser to advise the advertiser of status information about search listings of the advertiser.

[0253] One embodiment is implemented as a notification method in a computer database system. The method includes receiving a notification instruction from an owner associated with a search listing stored in the computer database system. The owner in one embodiment is an advertiser who is associated with a marketplace operator who owns, operates and maintains the computer database system. One particular embodiment of the method is practiced in conjunction with a database system accessible via the World Wide Web. In this exemplary embodiment, the search listing is one stored in the database search system and accessible by a search engine in response to a search query submitted by a third party searcher. Information about the search listing is presented to the searcher along with other search results. Factors such as the cost charged to the owner and the display rank of the search listing may be controlled by information contained in the notification instruction.

[0254] The notification instruction may be received at the computer database system in any suitable fashion. In one particular embodiment, the notification instruction is received when the owner or advertiser accesses a World Wide Web page of the marketplace operator and specifies one or more conditions and associated data states about which the owner should be automatically notified.

[0255] The notification method further includes monitoring conditions specified by the notification instruction for the search listing. Exemplary conditions include those specified herein and their equivalents. In particular, exemplary conditions include conditions related to a variable state of the search listing such as its associated cost and display rank. Other exemplary conditions include economic conditions such as an account balance of the owner or advertiser with the marketplace operator.

[0256] The notification method further includes sending a notification to the owner upon detection of a changed condition of the search listing. The notification may be communicated in any convenient way or combination of ways. The notification may include built-in information for responding to the notification, so that the condition can be corrected.

[0257] Another embodiment is implemented as a database search system. The database search system includes a database of search listings associated with advertisers. Each advertiser may initiate and maintain one or more search listings. The search listings may be searched to produce search results. The database search system further includes a processing system which sends a notification to an advertiser when a change condition of a search listing of the advertiser has occurred. The change condition may be specified by the

advertiser or may be a default or other operator-specified condition. In one embodiment, the change condition is identified by the advertiser and threshold values or limits are specified by the advertiser. The state of the condition is preferably automatically tracked or monitored until a change in the condition is detected. Subsequently, a notification is sent to the advertiser to alert of the change or some other action is taken by the system.

[0258] Another embodiment is implemented as a database search system which includes a database of search listings. Each search listing is associated with an advertiser. The database search system further includes a search engine. Still further, the database search system includes means responsive to condition specifying information from one or more advertisers for providing an indication to an advertiser when a specified condition of one or more search listings is satisfied.

[0259] The condition specifying information may be received from the one or more advertisers, may be a default or may be otherwise selected or specified or designated by an advertiser or others. The specified condition is tracked in a manner which may be appropriately chosen or specified depending on the condition and its nature. The indication may be of any sort or nature needed to communicate to the advertiser or some device or instrument associated with the advertiser that the specified condition is satisfied. The indication may be as simple as turning on or off some indicator or taking some action or failing to take some action. The presence or absence of the indicator or action may serve to communicate the condition to an advertiser. The indication may be more involved, such as a visual or audible communication conveyed to the advertiser with a built-in or automatic response.

[0260] Another embodiment is implemented as a database search system. The database search system includes in this embodiment a database of search listings. Each search listing is associated with a respective advertiser and each search listing includes a search term and a variable cost per click (CPC) or a variable display rank. The database search system in this embodiment further includes a search engine configured to identify search listings matching a search query received from a searcher. The matching search listings are preferably ordered in a search result list according to the display rank and the bid amount of the matching search listings. An agent is responsive to a condition definition from an advertiser to provide condition update information to the advertiser. The condition definition specifies a condition to be monitored. The condition update information, if present, specifies the circumstances under which the condition will be updated.

[0261] Another embodiment is implemented as a method for operating a database search system. In this embodiment, the method includes storing a plurality of search listings in a database. Each search listing is associated with an advertiser who gives economic value when a search listing is referred to a searcher. The method further includes determining a display position for associated search listings. In one example, the associated search listings are associated by common data, such as a search term or proximity to a search term. The display position may be determined in any appropriate way, from ways which are completely deterministic to ways which are completely random. The position determin-

ing way may be based on advertiser input or some other information. In one embodiment, each search listing is assigned a cost per click (CPC) and the display position is determined based on CPC, with the highest CPC listing for a search term being listed highest when that search term or a variant thereof is received. The method further includes receiving from an advertiser an indication of search listings for which the advertiser desires a notification of a display position change. The indication and the notification may be sent according to any suitable communication method any available, convenient communication channel.

[0262] The procedures illustrated in FIGS. 10-24 may be performed in software or hardware or any combination of these. In one embodiment, the procedures are initiated as software procedures running on the processing system 34 of the account management server 22 (FIG. 1). In other embodiments, the procedures may run on a separate machine with network access to the search listings database. The procedures together form an Auto Notification function.

[0263] The procedures illustrated in FIGS. 10-24 implement a notification method in a computer database system. The method includes acts such as receiving a notification instruction from an owner associated with a search listing stored in the computer database system, monitoring conditions specified in the notification instruction for the search listing, and sending a notification to the owner upon detection of a changed condition of the search listing.

[0264] In one embodiment, the computer database system is a pay for performance search system as described herein and includes a database of search listings and a search engine. The search listings are each associated with an advertiser or owner of the search listing. The search listings each include data such as a search term, a bid amount or maximum cost per clickthrough specified by the advertiser, a cost per clickthrough (CPC) and a rank or display rank. The CPC and the rank may be varied automatically depending on values specified by the advertiser and by other advertisers associated with search listings that include the same search term. For example, the system may automatically reduce the CPC of a listing to a minimum while still maintaining a specified rank. The search engine matches search terms or other portions of the search listings with a search query received from a searcher. The matching search listings are organized according to CPC and display rank and returned to the searcher. If a search listing is referred to the searcher, an economic value of an amount equal to the CPC is payable by the advertiser or owner, who may keep an account for this purpose. A referral of a search listing in this case might be an impression, such as including information about the search listing in the display results, a click through by the searcher, or some post-click through action by the searcher. This embodiment is exemplary only. The notification method may be applied to other types of database search systems as well for advising owners or others associated with listings in a database of a changed condition of a search listing.

[0265] One example of a changed condition which may be notified to the owner include a change of position of a search listing among the search results produced for a particular search term. Another example of a changed condition is when the CPC for a search listing reaches some value or range specified by the advertiser or owner. Another example

of a changed condition is when the owner's account balance falls below an owner-specified amount. Another example of a changed condition is when aggregate impressions for one or more of the advertiser's search listings exceed a specified number, or when aggregate clickthroughs exceed a specified number, or when the clickthrough rate over some specified time period exceeds a specified number. Another example of a changed condition occurs when the CPC of any search listing can be reduced without impacting its rank among other search listings for the same search term. Another example of a changed condition occurs when a search listing can be at an advertiser specified display rank for less than an advertiser specified CPC. Another example of a changed condition is when an advertisers average CPC across some collection of listings exceeds a predetermined threshold.

[0266] In one embodiment, the advertiser can select timing of the notifications sent by the system. Further, in one embodiment, the advertiser can specify the nature of the notification sent by the system, such as an electronic mail message, a facsimile, a page or a short or instant message. Still further, in one embodiment, the notification may include active links, inactive links or email responses specifying an action to be taken by the system to correct or resolve the notified condition.

[0267] FIG. 10 is a flow diagram illustrating one embodiment of a method for creating a new Auto Notification function. In accordance with the present embodiment, each advertiser can create a new Auto Notification function by specifying: 1) the condition type and the parameters for the condition type, 2) the notification time, 3) the notification mode, and 4) the notification action type.

[0268] Auto Notification functions are preferably implemented as one or more software agents implemented on a computer system such as the account management web server 22 of FIG. 1. When an Auto Notification function is created, the software routine is created by supervisory software operating on the system using information provided by an advertiser associated with the Auto Notification function and possibly standard or default information. In alternative embodiments, the Auto Notification function may be implemented incorporating dedicated hardware or software components or some combination of these. The system keeps track of all Auto Notification functions, and if any of the conditions tracked by the function becomes true, the system under control of the Auto Notification function records the details. The advertiser is notified immediately if the notification time is immediate. Otherwise all conditions are recorded and are later sent to the advertiser at the specified notification time. The notification is sent in the mode or modes specified by the advertiser, and each notification may include one or more action types to correct any of the undesirable conditions. Further, an advertiser may instruct the system to automatically correct any undesirable conditions.

[0269] The system also monitors all incoming corrective actions for previously sent notifications. If a corrective action is received, the system acts upon it to correct the condition of the notification, e.g., increasing the CPC of an advertiser's listing in order to restore it to the desirable rank.

[0270] The procedure illustrated in FIG. 10 accepts a new Auto Notification function from an advertiser. The procedure begins at block 1000. At block 1002, the condition type

for the Auto Notification function is selected. The condition type is specified by the advertiser and is specified for one or more search listings. The search listings are maintained in a search listing database, as described above in conjunction with FIG. 1. The condition type specifies one or more features of the search listing to be monitored by a software agent. At block 1004, the parameters for the conditions selected in block 1002 are specified. At block 1006, notification times are specified for the software agent. The notification times are associated with the conditions defined in blocks 1002, 1004. In the present embodiment, there are two choices when an advertiser can be notified. First, the advertiser can be notified immediately, or as soon as the specified condition becomes true. Second, the notifications can be aggregated over the specified time interval and, at the end of the interval, the advertiser is notified of all conditions that have become true during the interval. If no conditions have become true during the interval, then in one embodiment, no notification is sent. In other embodiments, a notification specifying no status change is sent.

[0271] At block 1008, notification modes for the selected conditions are specified. The modes may be specified by an advertiser or in any other suitable manner. In the present embodiment, there are several possible modes of communication of a notification to an advertiser. First, a notice may be sent to one or more electronic mail addresses specified by the advertiser. Second, a notice may be sent by an instant message account system, again as specified by the advertiser. Third, the notification may be sent by facsimile, faxed to a number prescribed, the advertiser. Fourth, the notice may be sent as a wireless page, as part of a paging system or in conjunction with a radiotelephone or other two-way communication system. Finally, the notification may be sent by telephone, either using an automated system to send and receive information from the advertiser or by means of an operator interacting with the advertiser.

[0272] At block 1010, the action types to be included with any notifications are specified by the advertiser. Possible action types include providing an active link embedded in the notification which, when clicked, allows the advertiser to correct the undesirable condition in a single click. In a second action type, the notification may be sent with inactive links, which are pointers to all locations where an advertiser can go to correct any undesirable conditions. A third action type is an electronic mail template which can be filled out by an advertiser who specifies what corrective actions are to be taken and returns the electronic mail to a specified address. Lastly, in the present embodiment, an action type may include provision of a telephone number which the advertiser can call to take corrective action. The process of initiating a new Auto Notification function ends at block 1012.

[0273] The method of FIG. 10 may be embodied in accordance with the pseudocode below.

[0274] Procedure New-Auto-Notification()

[0275] Select condition type;

[0276] Specify parameters for selected condition;

[0277] Specify notification times;

[0278] Specify notification modes;

[0279] Select action types to be included with any notifications;

[0280] End Procedure;

[0281] As described earlier, in the illustrated embodiment, there are nine types of conditions that an advertiser can select from:

[0282] 1. position: related to the position of a listing

[0283] 2. cost: related to the accumulated costs for some listings

[0284] 3. account-balance: related to the funds remaining in the advertiser's account.

[0285] 4. impressions: the number of impressions of some listings

[0286] 5. clicks: the number of clicks of some listings

[0287] 6. CTR: the click through rate of some listings

[0288] 7. CPC-too-high: if the CPC of a listing can be reduced without impacting its rank

[0289] 8. Average CPC too high: the average CPC, the total cost divided by the total clicks, is higher than some threshold.

[0290] 9. rank-CPC: related to the CPC for a given rank and term

[0291] Other conditions may be specified as well.

[0292] Each condition has a set of parameters for it. After an advertiser selects a condition type, he must specify the parameters for it. The parameters for each of the eight condition types were defined earlier. It is possible for the operator of the marketplace or pay for performance system to provide default values for some of the parameters, depending on the context in which the advertiser is interacting with the system.

[0293] The advertiser must also select the notification time(s). This can be "immediate" or "interval." Immediate notifications are sent to the advertiser as soon as the system detects that they are true. Interval notifications, on the other hand, are only sent periodically. The advertiser must specify the interval, e.g., daily. Every time a condition is detected to be true by the system, a log of the details is recorded. At the boundary of every interval the system gathers up all instances of the conditions that are or were true, and includes the details of these in the body of the notification. For example, for a daily interval, once a day the system will send a report of all the conditions that were true in the past 24 hours. The marketplace operator can define the boundaries of the interval (e.g., midnight for daily intervals).

[0294] The advertiser must also select one or more notification modes. Notifications can be sent to the advertiser in all the selected notification modes. In one embodiment, there are five notification modes:

[0295] 1. email: the advertiser must specify one or more e-mail address, which can have a default value.

[0296] 2. instant messaging: the advertiser must specify the instant-message address, which can have a default value.

[0297] 3. fax: the advertiser must specify the fax number, which can have a default value.

[0298] 4. page: the advertiser must specify the page number, which can have a default value.

[0299] 5. phone: the advertiser must specify the phone number, which can have a default value.

[0300] Other notification modes may be specified as well.

[0301] Finally, the advertiser may choose to select one or more notification action types for each notification mode selected earlier. Each notification mode has one or more action types that are applicable for use with it. The marketplace operator may automatically provide defaults for the applicable action types for each notification mode. In the illustrated embodiment, there are four notification action types:

[0302] 1. active links: these can be included in e-mail notifications and instant messaging notifications.

[0303] 2. inactive links: these can be included in all notification modes.

[0304] 3. e-mail: these can be included in all notification modes.

[0305] 4. phone: these can be included in all notification modes.

[0306] Other notification action types may be specified as well.

[0307] Thus, FIG. 10 shows one method for initializing a new Auto Notification function. The advertiser can also cancel an existing Auto Notification function at any time.

[0308] FIG. 11 is a full diagram illustrating operation of a software agent to provide an Auto Notification function, monitoring conditions specified by an advertiser in accordance with the process of FIG. 10. The method of FIG. 11 begins at block 1100.

[0309] At block 1102, a variable new-true-condition is initialized to be an empty list. At block 1104, the software agent waits for an event that may make a condition true. Such events include a change in ranking due to bid changes submitted by advertisers and clickthroughs by searchers which may change a monitored clickthrough condition or the clickthrough rate, etc. At block 1106, a process called check-all-conditions is initialized. This process will be described further below in conjunction with FIG. 12.

[0310] At block 1108, a variable X is set equal to the next element in the list new-true-conditions. At block 1110, a test is performed to determine if the variable X is empty or stores no data. If X is empty, there are no more elements of the list new-true-conditions to be processed and control returns to block 1102. If X is not empty, control proceeds to block 1112.

[0311] At block 1112, it is determined if X has an auto-correction defined for it by the advertiser. If so, control proceeds to block 1114, where the automatic corrective action specified by the advertiser is performed by the system. The corrective action can be any action specified by the advertiser. If, at block 1112, the condition associated with the variable X does not have an auto-correction, control proceeds to block 1116.

[0312] At block 1116, it is determined if X has an immediate notification. If so, control proceeds to block 1120, a procedure notify-immediately is initiated to send a notification of the condition to the advertiser. One embodiment of this procedure will be described further below in connection with FIG. 21. If, at block 1116, the condition associated with variable X does not have an immediate notification, at block 1118, the condition associated with variable X is recorded and control returns to block 1108.

[0313] In the procedure monitor-conditions illustrated in FIG. 11, the system continually monitors its state to see if any conditions have become true. Only the transition from a condition being false to a condition being true is relevant. For example, a condition event is recorded when an advertiser's account balance falls below the set threshold. If the balance further decreases, this is not recorded as a separate instance of the condition being true.

[0314] The method of FIG. 11 may be embodied in accordance with the pseudocode below.

Procedure monitor-conditions ()

```

Loop
Assign new-true-conditions = empty list;
Wait for an event that may make a condition true;
Check-all-conditions;
Loop x over new-true-conditions
  If x has an immediate notification time
    Notify-immediately (x);
  Else
    Record condition x;
  End If;
End Loop;
End Loop;
End Procedure;

```

[0315] Checking if any conditions have become true involves checking the eight different condition types: FIG. 12 illustrates one embodiment of the procedure check-all-conditions, implemented at block 1106 of FIG. 11. Checking if any conditions have become true involves checking 8 different condition types. This is illustrated in FIG. 12. The method begins at block 1200.

[0316] At block 1202, all position conditions are checked. At block 1204 all cost conditions are checked. At block 1206 all account balance conditions are checked. At block 1208 all impressions conditions are checked. At block 1210, all clicks conditions are checked. At block 1212 all clickthrough rate (CTR) conditions are checked. At block 1214 all CPC-too-high conditions are checked. At block 1216, all average CPC-too-high conditions are checked. At block 1218, all rank CPC conditions are checked. The method ends at block 1220. In alternative embodiments, only one or more subsets of these conditions may be checked at any given time. Alternatively, if not all condition types are supported in an embodiment, some checks may be omitted. Particular embodiments of the methods for checking the conditions specified in FIG. 12 are illustrated in greater detail in FIG. 13-21.

[0317] The method of FIG. 12 may be embodied in accordance with the pseudocode below.

- [0318] Procedure Check-all-conditions()
- [0319] Check-position-conditions;
 - [0320] Check-cost-conditions;
 - [0321] Check-account-balance-conditions;
 - [0322] Check-impressions-conditions;
 - [0323] Check-clicks-conditions;
 - [0324] Check-CTR-conditions;
 - [0325] Check-CPC-too-high-conditions;
 - [0326] Check-average-CPC-too-high-conditions;
 - [0327] Check-rank-CPC-conditions;
 - [0328] End Procedure;

[0329] FIG. 13 is a flow diagram illustrating a procedure to check if any "position" conditions have become true. The method begins at block 1300.

[0330] At block 1302, variable L is set equal to search listings with monitor positions whose position has changed. The identity of these search listings may be determined in any suitable manner. At block 1304, the variable X is incremented to be the next element in the list L. At block 1306, a test is performed to determine if variable X is empty. If so, the end of the list contained in variable L has been reached and the procedure ends at block 1308. If not, at block 1310, it is determined if the position condition associated with the listing in variable X is currently true and if the last position condition associated with this variable is false. This is determined by comparing the current position condition associated with the search listing indicated by the variable X with a stored last position condition for this variable. If the test of block 1310 produces a true or yes response, at block 1312, the variable last position condition for the search listing X is reset equal to a true value and, in block 1314, the position condition for the variable X is adjoined to the list of new-true-conditions. Control then returns to block 1304 to select the next element in the list L.

[0331] If, at block 1310, the test produced a negative or false response, at block 1316 another test is performed to determine if the position condition for variable X is false and the last position condition for variable X is true. If not, control returns to block 1304. If so, at block 1318, the last position condition for the search listing associated with the variable X is set equal to a false value. Control then returns to block 1304.

[0332] The method of FIG. 13 may be embodied in accordance with the pseudocode below.

```

Procedure Check-position-conditions ( )
Assign L = listings with monitored position whose position has
changed;
Loop x over all elements in L
  If position-condition(x) = true and
  last-position-condition(x) = false
    Assign last-position-condition(x) = true;
    Adjoin position-condition(x) to new-true-conditions;
  Else If position-condition(x) = false and
  last-position-condition(x) = true

```

-continued

```

Procedure Check-position-conditions ( )
Assign last-position-condition(x) = false;
End If;
End Loop;
End Procedure;

```

[0333] Whenever a position condition is first created, its "last-position-condition" is automatically initialized to be false, and its position is treated as if it has changed-this permits the condition to be tested immediately.

[0334] FIG. 14 illustrates one method for checking cost conditions. The procedure begins at block 1400. At block 1402, a variable L is initialized with all search listings with monitored cost and new cost for (CPC) charges. At block 1404, a variable X is assigned equal to the next element in the list L. At block 1406, it is determined if the variable X is empty. If so, the procedure ends at block 1407.

[0335] Otherwise, at block 1408, it is determined if the cost-condition for the search listing associated with the variable X is true and the last-cost-condition for the search listing associated with the variable X was false. If so, at block 1410, the last-cost-condition for the search listing is set equal to true. At block 1412, the cost-condition for the search listing associated with the variable X is adjoined to a list of new-true-conditions. Control returns to block 1404.

[0336] If, at block 1408, the cost-condition for the search listing had not changed from a previous false to a current true, it is determined if the cost-condition for the search listing associated with the variable X is false and the last cost-condition for the search listing was true. If so, the state of the last-cost-condition for the search listing associated with the variable X is set equal to false at block 1416. Control then returns to block 1404.

[0337] The method of FIG. 14 may be embodied in accordance with the pseudocode below.

```

Procedure check-cost-conditions ( )
Assign L = listings with monitored cost with new CPC charges;
Loop x over all elements in L
  If cost-condition(x) = true and
  last-cost-condition(x) = false
    Assign last-cost-condition(x) = true;
    Adjoin cost-condition(x) to new-true-conditions;
  Else If cost-condition(x) = false and
  last-cost-condition(x) = true
    Assign last-cost-condition(x) = false;
  End If;
End Loop;
End Procedure;

```

[0338] Whenever a cost condition is first created, its "last-cost-condition" is automatically initialized to be false, and its CPC is treated as if it has changed-this permits the condition to be tested immediately. Note that checking a cost condition requires checking the accumulated costs for all listings in the condition since the last interval. We ignore all costs at time points earlier than the most recent advertiser-defined interval for this condition. For example, if the interval is "daily", then all costs for the previous day are ignored (the marketplace operator can define the boundary for the intervals).

[0339] FIG. 15 illustrates one embodiment of a method for checking account balance conditions. The method begins at block 1500. At block 1502, a variable L is assigned to contain a list of advertisers with monitored account balances with new charges. At block 1504, a variable X is incremented to contain the next element in the list L. At block 1506, it is determined if variable X is empty. If so, the procedure ends at block 1508.

[0340] Otherwise, at block 1510, it is determined if the account-balance-condition for the search listing associated with the variable X is true and if the last-account-balance-condition for the search listing was false. If so, at block 1512, the variable last-account-balance-condition for the search listing is set equal to true. At block 1514, the account-balance-condition for the search listing is adjoined to a list of new-true-conditions. Control then returns to block 1504.

[0341] If the result of the test at block 1510 was negative, at block 1516, a test is performed to determine if the account-balance-condition for the search listing associated with the variable X is now false and the last-account-balance-condition for the search listing was true. If so, at block 1518, a variable last-account-balance-condition for the search listing associated with the variable X is set equal to false. Control returns to block 1504 to select the next element in the list L.

[0342] The method of FIG. 15 may be embodied in accordance with the pseudocode below.

```
Procedure check-account-balance-conditions ( )
```

```
Assign L = advertisers with monitored account balance w new charges;
Loop x over all elements in L
  If account-balance-condition(x) = true and
    last-account-balance-condition(x) = false
    Assign last-account-balance-condition(x) = true;
    Adjoin account-balance-condition(x) to new-true-conditions;
  Else If account-balance-condition(x) = false and
    last-account-balance-condition(x) = true
    Assign last-account-balance-condition(x) = false;
  End If;
End Loop;
End Procedure;
```

[0343] In the illustrated embodiment, whenever an account balance condition is first created, its “last-account-balance-condition” is automatically initialized to be false, and it is treated as if it has new charges-this permits the condition to be tested immediately.

[0344] FIG. 16 illustrates a method for checking impressions conditions. The method begins at block 1600. At block 1602, the list variable L is initialized with all listings with monitored impressions having new impressions. At block 1604, the variable X is incremented to point to the next element in the list L. At block 1606, it is determined if the search listing pointed to by the variable X is empty. If so, the procedure terminates at block 1608. Otherwise, at block 1610, it is determined if the impressions-condition for the search listing designated by variable X is true and if the last-impression-condition for the search listing associated with the variable X was false. If so, at block 1612, the variable-last-impressions-condition for the search listing is

set equal to true. At block 1614, the value of the variable-impressions-condition for the search listing is adjoined to the list new-true-conditions. Control returns to block 1604 to select a next element in the list L.

[0345] If, at block 1610, the test return a negative result, at block 1616 it is determined if the variable impressions-condition for the search listing associated with the variable X has a false value and if the variable-last-impressions-condition for the search listing has a true value. If so, then at block 1618, the variable-last-impressions-condition for the search listing is assigned a value of false and control returns to block 1604.

[0346] The method of FIG. 16 may be embodied in accordance with the pseudocode below.

```
Procedure check-impressions-conditions ( )
```

```
Assign L = listings with monitored impressions with new impressions;
Loop x over all elements in L
  If impressions-condition(x) = true and
    last-impressions-condition(x) = false
    Assign last-impressions-condition(x) = true;
    Adjoin impressions-condition(x) to new-true-conditions;
  Else If impressions-condition(x) = false and
    last-impressions-condition(x) = true
    Assign last-impressions-condition(x) = false;
  End If;
End Loop;
End Procedure;
```

[0347] Whenever an impressions condition is first created, its value of last-impressions-condition is automatically initialized to be false, and it is treated as if it has a new impression. This permits the condition to be tested immediately. Note that checking an impressions condition requires checking the accumulated impressions of all the listings in the condition. We ignore all impressions that are earlier than the most recent advertiser-defined interval for this condition. For example, if the interval is “daily”, then all impressions for the previous day are ignored (the marketplace operator can define the boundary for the intervals).

[0348] FIG. 17 illustrates one embodiment of a method for checking clicks conditions. The method begins at block 1700. At block 1710, a list variable L is filled with listings with monitored clicks having new clicks. At block 1712, a variable X is initialized or incremented to contain the next element in the list variable L. At block 1714, it is determined if the variable X is empty. If so, at block 1716, the procedure ends. If not, at block 1718, it is determined if the clicks-condition for the search listing associated with the variable X has a value true and the variable last-clicks-condition for the search listing associated with the variable X had a value false. If so, at block 1720, the variable last-clicks-condition for the search listing is set equal to a value true. At block 1722, the contents of the variable clicks-condition for the search listing are adjoined to the list of new-true-conditions. Control returns to block 1712.

[0349] If the test at block 1718 had a negative result, at block 1724, it is determined if the clicks-condition for the search listing has a variable false and if the variable last-clicks-condition for the search listing had a variable true. If

so, at block 1726 the variable last-clicks-condition for the search listing is set equal to a value false. Control then returns to block 1712.

[0350] The method of FIG. 17 may be embodied in accordance with the pseudocode below.

```

Procedure check-clicks-conditions ( )
Assign L = listings with monitored clicks with new clicks;
Loop x over all elements in L
  If clicks-condition(x) = true and
    last-clicks-condition(x) = false
    Assign last-clicks-condition(x) = true;
    Adjoin clicks-condition(x) to new-true-conditions;
  Else If clicks-condition(x) = false and
    last-clicks-condition(x) = true
    Assign last-clicks-condition(x) = false;
  End If;
End Loop;
End Procedure;
    
```

[0351] Whenever a clicks condition is first created, its value of last-clicks-condition is automatically initialized to be false. It is treated as if it has a new click. This permits the condition to be tested immediately. Note that checking a clicks condition requires checking the accumulated clicks of all the listings in the condition. We ignore all clicks that are earlier than the most recent advertiser-defined interval for this condition. For example, if the interval is “daily”, then all clicks for the previous day are ignored.

[0352] FIG. 18 is a flow diagram illustrating a method for checking clickthrough rate (CTR) conditions. Clickthrough rate is the number of clickthroughs for a search listing in a specified time period divided by the specified time period, such as clicks per hour or clicks per day. The method begins at block 1800.

[0353] At block 1802, a list variable L is initialized with all search listings having monitored clickthrough rate and with new clicks. At block 1804, a variable X is initialized to point to the next element in the list variable L. At block 1806, it is determined if the variable X is empty. If so, at block 1808 the method ends. If not, at block 1810, it is determined if the variable CTR-condition for the search listing designated by the variable X is true and if the last-CTR-condition for the search listing had a value false. If so, at block 1812, the variable last-CTR-condition for the search listing is set equal to a value true. At block 1814, the contents of the variable CTR-condition for the search listing is adjoined to the list of new-true-conditions and control returns to block 1804.

[0354] If, at block 1810, the test produced a negative result, at block 1816 it is determined if the variable CTR-condition for the search listing associated with the variable X has a value false and if the variable last-CTR-condition for the search listing had a value true. If so, at block 1818, the variable last-CTR-condition for the search listing is assigned a value false and control then returns to block 1804.

[0355] The method of FIG. 18 may be embodied in accordance with the pseudocode below.

```

Procedure check-CTR-conditions ( )
Assign L = listings with monitored CTR with new clicks;
Loop x over all elements in L
  If CTR-condition(x) = true and
    last-CTR-condition(x) = false
    Assign last-CTR-condition(x) = true;
    Adjoin CTR-condition(x) to new-true-conditions;
  Else If CTR-condition(x) = false and
    last-CTR-condition(x) = true
    Assign last-CTR-condition(x) = false;
  End If;
End Loop;
End Procedure;
    
```

[0356] Whenever a CTR condition is first created, its value of last-CTR-condition is automatically initialized to be false. It is treated as if it has a new click. This permits the condition to be tested immediately. Note that checking a CTR condition requires checking the accumulated impressions and clicks for all the listings in the condition. We ignore all impressions and clicks that are earlier than the most recent advertiser-defined interval for this condition. For example, if the interval is “daily”, then all impressions and clicks for the previous day are ignored. The marketplace operator may require a minimum number of impressions before considering the CTR to be valid.

[0357] FIG. 19 is a flow diagram illustrating a method for checking CPC-too-high conditions. These are conditions where the cost per click is higher than necessary. The procedure begins at block 1900.

[0358] At block 1902, a list variable L is initialized with all search listings having monitored CPC-too-high where the listing below has a new CPC. The listing below has a new CPC if the CPC of the listing below is changed, or if a new listing is inserted below, or if the previous listing below is removed. At block 1904, a variable X is set to point to the next element in the list variable L. At block 1906, it is determined if the variable X is empty, indicating the end of the list L has been reached. If X is empty, at block 1908 the procedure ends. Otherwise, at block 1910, a variable C is set equal to the condition of X. At block 1912, it is determined if there is no listing below X. If not, at block 1916, the variable lower-CPC is set equal to the cost per click for the search listing immediately below the search listing indicated by the variable X. If there is no a search listing below the listing indicated by the variable X, at block 1914, the variable lower-CPC is set equal to the difference between the minimum cost per click for the system and a CPC threshold for the search listing. At block 1918, it is determined if the cost per click for the search listing is greater than the value of lower-CPC plus CPC threshold for the variable C and if the value of the variable last-CPC-too-high-condition for the variable C is false. If so, at block 1920, the variable last-CPC-too-high-condition is set equal to true. At block 1922, the contents of the variable CPC-too-high-condition are adjoined to the list of new-true-condition and control returns to block 1904.

[0359] If at block 1918 the test produced a negative result, at block 1924, it is determined if the CPC for the search

listing is less than or equal to the lower-CPC plus the CPC-threshold and if the value of the variable last-CPC-too-high-condition is equal to true. If so, at block 1926, the variable last-CPC-too-high-condition is set equal to a value of false. Control then returns to block 1904. The method of FIG. 19 may be embodied in accordance with the pseudocode below.

```

Procedure check-CPC-too-high-conditions( )
Assign L = listings with CPC-too-high monitor where the listing below
has a new CPC;
Loop x over all elements in L
  Assign c = condition of x;
  If no listing below x
    Assign lower-CPC = Min-CPC - CPC-threshold(c);
  Else
    Assign lower-CPC = CPC(listing below x);
  End If;
  If CPC(x) > lower-CPC + CPC-threshold(c) and
  Last-CPC-too-high-condition(c) = false
    Assign last-CPC-too-high-condition(c) = true;
    Adjoin CPC-too-high-condition(c) to new-true-conditions;
  Else if CPC(x) lower-CPC + CPC-threshold(c) and
  Last-CPC-too-high-condition(c) = true
    Assign last-CPC-too-high-condition(c) = false;
  End If;
End Loop;
End Procedure;

```

[0360] Whenever a CPC-too-high condition is first created, its value of last-CPC-too-high-condition is automatically initialized to be false and it is treated as if the listing directly below it has a new CPC. This permits the condition to be tested immediately. Min-CPC is the minimum CPC for all listings, which is determined by the marketplace operator and in one example is \$0.01. Every CPC-too-high condition has an advertiser defined threshold. This threshold is the difference between the CPC of the listing and the CPC of the listing below must be greater than this threshold for the condition to be true. If there is no listing below, we check if the CPC of the listing is higher than the minimum CPC, and alternatively we could check if the CPC is the threshold above the minimum CPC.

[0361] FIG. 20 is a flow diagram illustrating a method for checking average CPC too high conditions. The average CPC for a set of listings is the aggregate cost for the set of listings over an interval divided by the aggregate clicks for the set of listings for the same interval. The method begins at block 2000.

[0362] At block 2002, a list variable L is initialized with the list of conditions having monitored average CPC too high and with new clicks. Each such condition has an associated set of listings whose average CPC is being monitored. At block 2004, a variable X is initialized to point to the next element in the list variable L. At block 2006, it is determined if the variable X is empty. If so, at block 2008 the method ends. If not, at block 2010, it is determined if the variable average-CPC-too-high-condition for the condition designated by the variable X is true and if the last-average-CPC-too-high-condition for the condition had a value false. If so, at block 2012, the variable last-average-CPC-too-high-condition for the condition is set equal to a value true. At block 2014, the contents of the variable average-CPC-too-

high-condition for the search listing is adjoined to the list of new-true-conditions and control returns to block 2004.

[0363] If, at block 2010, the test produced a negative result, at block 2016 it is determined if the variable average-CPC-too-high-condition for the condition associated with the variable X has a value false and if the variable last-average-CPC-too-high-condition for the search listing had a value true. If so, at block 2018, the variable last-average-CPC-too-high-condition for the condition is assigned a value false and control then returns to block 2004.

[0364] The method of FIG. 20 may be embodied in accordance with the pseudocode below.

```

Procedure check-average-CPC-too-high-conditions( )
Assign L = all conditions with monitored CPC too high with new
clicks;
Loop x over all elements in L
  If average-CPC-too-high-condition(x) = true and
  last-average-CPC-too-high-condition(x) = false
    Assign last-average-CPC-too-high-condition(x) = true;
    Adjoin average-CPC-too-high-condition(x) to new-true-
conditions;
  Else If average-CPC-too-high-condition(x) = false and
  last-average-CPC-too-high-condition(x) = true
    Assign last-average-CPC-too-high-condition(x) = false;
  End If;
End Loop;
End Procedure;

```

[0365] Whenever an average CPC too high condition is first created, its value of last-average-CPC-too-high-condition is automatically initialized to be false. It is treated as if it has a new click. This permits the condition to be tested immediately. Note that checking an average CPC too high condition requires checking the accumulated clicks and costs for all the listings in the condition. We ignore all clicks and costs that are earlier than the most recent advertiser-defined interval for this condition. For example, if the interval is "daily", then all clicks and costs for the previous day are ignored. The marketplace operator may require a minimum number of clicks before considering the average CPC to be valid.

[0366] FIG. 21 is a flow diagram illustrating a method for checking rank CPC conditions. The method begin at block 2100. At block 2102, a variable L is assigned equal to all search terms with monitored rank CPC where the cost per click of a monitored rank has changed. At block 2104, a variable X is set to point to the next element in the list L. At block 2106, it is determined if the variable X is empty. If so, this indicates that the end of the list has been reached and the method ends at block 2108.

[0367] Otherwise, at block 2110, a variable M is set equal to all ranks of search terms indicated by the variable X that are monitored and have new CPCs. At block 2112 a variable Y is initialized to point to the next element in the list M. At block 2114, it is determined if variable Y is empty. If so, control returns to block 2104 to select the next element in list L. Otherwise, at block 2116, a variable N is set equal to all rank CPC conditions for the search term indicated by the variable X and the rank indicated by the variable Y. At block 2118, variable Z is set equal to the next element in the list N. At block 2120, it is determined if the variable Z is empty.

If so, control returns to block 2104. Otherwise, at block 2122 it is determined if the value of the variable rank-CPC-condition for the condition indicated by the variable Z has a value true and if the variable last-rank-CPC-condition for the condition indicated by variable Z had a value false. If so, at block 2124, the variable last-rank-CPC-condition for the condition is set equal to true. At block 2126, the contents of the variable rank-CPC-condition for the condition are adjoined to the list of new-true-conditions. Control then returns to block 2104.

[0368] If at block 2122 a negative result was produced, at block 2128 a test determines whether the variable rank-CPC-condition for the condition is false and the variable last-rank-CPC-condition for the condition is true. If so, at block 2130, the variable last-CPC-too-high-condition is set equal to a value false. Control then returns to block 2104.

[0369] The method of FIG. 21 may be embodied in accordance with the pseudocode below.

```

Procedure check-rank-CPC-conditions( )
Assign L = terms with a rank-CPC monitor where the CPC of a monitored
rank has changed;
Loop x over all elements in L
  Loop x over all ranks of x that are monitored and have new CPCs
    Loop z over all rank-CPC conditions for term x and rank y
      If rank-CPC-condition(z) = true and
        Last-rank-CPC-condition(z) = false
          Assign last-rank-CPC-condition(z) = true;
          Adjoin rank-CPC-condition (z) to
            new-true-conditions;
      Else if rank-CPC-condition(z) = false and
        Last-rank-CPC-condition(z) = true
          Assign last-rank-CPC-condition(z) = false;
      End If;
    End Loop;
  End Loop;
End Loop;
End Procedure;

```

[0370] Whenever a rank-CPC condition is first created, its value of last-rank-CPC-condition is automatically initialized to be false and it is treated as if the CPC for the monitored rank has changed. This permits the condition to be tested immediately. Every rank-CPC condition has an advertiser defined threshold. The condition is true if the CPC for the rank is less than or equal to the threshold.

[0371] The procedure “notify-immediately” sends a message to the advertiser with the details of the current condition that has become true. The procedure first selects all the notification modes selected by the advertiser. It next selects the action types. The advertiser can select which action type(s) he prefers. Some action types may not be available with all notification modes, e.g., the marketplace operator may only provide “active-links” in e-mails and instant-messages. Also, some conditions may not have any automatic corrective actions (e.g., CTR). The procedure sends a notice to the advertiser in each communication mode, where each message in a particular communication mode possibly includes a set of corrective actions:

[0372] FIG. 22 is a flow diagram illustrating one embodiment of the procedure notify-immediately. The procedure starts at block 2200. At block 2202, the variable L is

assigned equal to a list of all notification modes for the condition passed to the procedure which has become true. At block 2204, the variable X is initialized to be the next element in the list L. At block 2206, a test is performed to determine if the variable X is empty. If so, at block 2208, the procedure ends, as all elements of the list L have been processed.

[0373] If the variable X is not empty, at block 2210, the variable A is set equal to all action types for mode X necessary to correct the condition. At block 2212, a procedure send-notification is called, passing as parameters the condition which is true, the variable X and the variable Y. After processing of this procedure, control returns to block 2204 to select the next element in list L.

[0374] The method of FIG. 22 may be embodied in accordance with the pseudocode below.

```

Procedure Notify-immediately(condition)
Loop x over all notification modes for condition
  Assign y = all action types for mode x to correct condition;
  Send-notification (condition,x,y);
End Loop;
End Procedure;

```

[0375] The following is a list of the conditions that can have corrective actions included in a notification message. An advertiser can either accept the suggested corrective action in the message, or he can ignore it. The corrective actions are steps that can be taken automatically on behalf of the advertiser to ensure that the condition is no longer true. Note that a corrective action is not applicable if the condition is no longer true:

[0376] 1. position: it may be possible to correct a position condition by changing the CPC of a listing. For example, if the condition “Listing L₁ is not at rank 3” is true, then it may be possible to return L₁ to rank 3 by increasing the CPC if L₁ is at rank worse than 3, or by decreasing the CPC if L₁ is at a rank better than 3.

[0377] A possible corrective action is “Adjust my CPC to return listing L₁ to rank 3”.

[0378] 2. account balance: it may be possible to correct an account balance condition by adding more funds to the account. The advertiser may select the additional amount to add.

[0379] A possible corrective action is “Add \$200.00 to my account balance from my credit card”.

[0380] 3. CPC too high: it may be possible to correct a CPC too high condition by reducing the CPC to the minimum required to maintain the current rank.

[0381] A possible corrective action is “Reduce the CPC of listing L₁ to the minimum required for its current rank”.

[0382] The procedure notify-interval sends a message to the advertiser with the details of all the condition that have become true in the last interval (the duration of the interval is specified by the advertiser). All the conditions that became

true in the interval are gathered together in one message. The procedure selects all the notification modes selected by the advertiser. It next selects the action types. The advertiser can select which action type(s) he prefers. Some action types may not be available with all notification modes, e.g., the marketplace operator may only provide “active-links” in e-mails and instant-messages. Also, some conditions may not have any automatic corrective actions (e.g., CTR).

[0383] If there is more than one instance of a condition, then it is only possible to have a corrective action for the most recent instance. It is possible that a condition that was recorded earlier is no longer true, in which case it will not have any corrective action associated with it. The procedure sends a notice to the advertiser in each communication mode, where each message in a particular communication mode possibly includes a set of corrective actions:

[0384] FIG. 23 is a flow diagram illustrating a procedure notify-interval. The procedure begins at block 2300. At block 2302, the procedure pauses to wait for a notification time for any advertiser. As noted above, information about changed conditions can be communicated by the system to an advertiser according to any schedule specified by the advertiser. The operation at block 2302 is performed according to the advertiser specified schedule.

[0385] At block 2304, the list variable L is assigned equal to all advertisers with a current notification time. That is, all advertisers who have specified a notification schedule which matches the current time. At block 2306, the looping variable X is assigned equal to the next element in the list variable L. At block 2308, X is tested to determine if the variable X is empty. If so, control returns to block 2302 to await a next notification time. If the variable X is not empty, control proceeds to block 2310.

[0386] At block 2310, the variable M is assigned equal to all auto notification conditions for the advertiser specified by the variable X for the current time which have recorded conditions. At block 2312, a looping variable Y is set equal to the next element in the list M. At block 2314, it is determined if the variable Y is empty. If so, control returns to block 2306 to select the next variable X in the list L. If variable Y is not empty, control proceeds to block 2316.

[0387] At block 2316, the variable C is assigned equal to all recorded instances of the condition whose value is stored in variable Y. At block 2318, the variable T is assigned equal to the most recent condition in variable C.

[0388] That is, the conditions and their associated time stamps are sorted or otherwise examined to determine a most recently occurring condition. This condition is loaded into the variable T. At block 2320, the variable M is set equal to all notification modes for the condition whose value is stored in variable Y.

[0389] At block 2322, the variable Z is incremented to point to the next element in the list N. At block 2324, it is determined if the looping variable Z is empty. If so, control returns to block 2306. Otherwise, at block 2326, the variable A is set equal to all action types for the mode stored in the variable Z, which may be specified by an advertiser to correct the condition specified by the variable T. At block 2328, a procedure send-notification is called, passing as parameters the variables C, Z, and A. Following execution of this procedure, control returns to block 2306 to select the next advertiser selected.

[0390] The method of FIG. 23 may be embodied in accordance with the pseudocode below.

```

Procedure Notify-interval()
Loop
Wait for a notification time for any advertiser;
Loop x over all advertisers with current notification time
  Loop y over all auto-notification for x for current
  time with recorded conditions
    Assign c = all recorded instances of condition y;
    Assign t = most recent condition in c;
    Loop z over all notification modes for y
      Assign a = all action types for mode z to
      correct condition t;
      Send-notification (c, z, a);
    End Loop;
  End Loop;
End Loop;
End Loop;
End Procedure;

```

[0391] The procedure handle-actions handles incoming actions. A message sent to an advertiser can include an action to correct an undesirable condition. The advertiser can choose to ignore the suggested action, or the advertiser can accept the suggested corrective action, in which case the system must act upon it.

[0392] The procedure handle-actions also handles a special advertiser action “tell-me-now.” An advertiser can create an auto-notification with a notification time that is an interval. However, at any time the advertiser can send a tell-me-now action, which instructs the system to send all notifications immediately. All notifications for the interval are still sent at the end of the interval, even if the advertiser sends a tell-me-now action. For example, an advertiser may have set up a notification interval “weekly, on Fridays at 3:45 p.m.” On Wednesday, the advertiser can send a “tell-me-now” action, which results in the advertiser receiving all the notifications recorded to date. The weekly notifications on Friday at 3:45 p.m. is still sent.

[0393] The procedure handle-actions also handles the actions “mute” and “un-mute”. The mute action allows the advertiser to stop all notifications—conditions are still recorded, but they are not sent (neither immediately nor at the specified intervals). Instead, the notifications are recorded for future transmission. The un-mute action re-enables the notification of conditions. All past-due notifications are immediately sent (e.g., for immediate notifications and notifications for past intervals). Other notifications will be sent at the end of the interval.

[0394] A separate handler is required for each notification action type (active links, inactive links, e-mail, and phone in the illustrated embodiment). Each of the action types has a procedure of the form below:

[0395] FIG. 24 is a flow diagram illustrating one embodiment of the procedure handle-actions. The procedure begins at block 2400. At block 2402, the procedure pauses, awaiting an incoming action. The action corresponds to a correction or other variation specified by an advertiser to correct an undesirable condition and the search listings maintained by the advertiser. After an action has been received, at block 2404 the action parameters are extracted from the received action. For example, the action may be transmitted as one or

more TCP/IP packets, containing instructions and data for correcting the undesirable condition. These instructions and data are extracted from the packets received from the advertiser.

[0396] At block 2406, the variable C is assigned equal to the value corresponding to the condition to be corrected by the received action. At block 2408, it is determined if the condition associated with the values stored in the variable C is still true. If not, at block 2410 a notification is sent to the advertiser associated with the condition that the condition is no longer true. If the condition is still true, at block 2412, a corrective action is executed. The corrective action may be any step or group of steps necessary to change or correct or otherwise modify the condition specified by the advertiser. After execution of the corrective action, at block 2414, it is determined if the action succeeded. That is, it is determined if the desired correction was obtained. If not, the advertiser is notified of the failure to correct the specified action at block 2416. If the action did succeed, at block 2418 the advertiser is notified of the success. Control then returns to block 2402 to await a next incoming action.

[0397] The method of FIG. 24 may be embodied in accordance with the pseudocode below.

```

Procedure Handle-actions()
  Loop
  Wait for an incoming action;
  Extract action parameters;
  Assign c = condition of action;
  If c is still true
    Execute corrective action;
    If successful
      Notify advertiser(c) of success;
    Else
      Notify advertiser(c) of failure;
    End If;
  Else
    Notify advertiser (c) that c is no longer true;
  End If;
  End Loop;
  End Procedure;

```

[0398] From the foregoing, it can be seen that the present embodiments provide a method and apparatus for advertisers associated with a pay for performance database to manage their listings more effectively. Procedures are provided to specify automatic software agents which monitor the search listings of an advertiser and provide notifications of the occurrence of specified conditions. Notifications of the conditions may be provided to the advertiser in any of a number of convenient channels, such as email or page or facsimile. The notifications may include action types built right in to the notifications so that the advertiser can respond rapidly and conveniently. This increases advertiser convenience, allows more rapid response to changing conditions, and frees up personnel who have heretofore been assigned to monitoring the status of search listings for the advertiser. The features may be provided at minimal expense to the pay for performance system operator and the advertisers.

[0399] While a particular embodiment of the present invention has been shown and described, modifications may be made. It is therefore intended in the appended claims to cover such changes and modifications which follow in the true spirit and scope of the invention.

1. A notification method in a computer database system comprising:

receiving a notification instruction from an owner associated with a search listing stored in the computer database system;

monitoring conditions specified by the notification instruction for the search listing; and

sending a notification to the owner upon detection of a changed condition of the search listing.

2. The notification method of claim 1 wherein receiving the notification instruction comprises:

receiving identification information for one or more search listings for which the associated owner desires a notification.

3. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about a change in position among search results for the search listing when the search listing is referred with other search listings forming the search results to a searcher in response to a search query from the searcher.

4. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about a value of cost per clickthrough for the search listing, the cost per clickthrough being an economic value payable by the owner when the search listing is referred to a searcher in response to a search query from the searcher.

5. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about an account balance for an account associated with the advertiser.

6. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about aggregate impressions for identified search listings.

7. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about aggregate clickthroughs for identified search listings.

8. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about aggregate clickthrough rate for identified search listings.

9. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information for notification about search listings having a cost per clickthrough which is reducible without affecting an advertiser specified display rank among search results when the search listing is referred among search results to a searcher in response to a search query from the searcher.

10. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information including a specified cost per clickthrough and a specified display rank for notification when the identified search listings can be at the specified display rank among search results presented to a searcher in response to a search query from the search for less than the specified cost per clickthrough.

11. The notification method of claim 2 wherein receiving the identification information comprises:

receiving identification information about an average cost per clickthrough for two or more search listings.

12. The notification method of claim 1 wherein receiving the notification instruction comprises:

receiving information defining notification times for sending the notification.

13. The notification method of claim 12 wherein receiving information defining notification times comprises one of:

receiving identification of search listings for sending an immediate notification, and

receiving identification of search listings for sending an interval notification.

14. The notification method of claim 1 wherein receiving the notification instruction comprises:

receiving an indication of a notification mode.

15. The notification method of claim 14 wherein the notification mode is selected from the group including: electronic mail, instant messaging, facsimile, paging and telephone voice call.

16. The notification method of claim 1 wherein receiving the notification instruction comprises:

receiving an indication of one or more action types to include with the notification.

17. The notification method of claim 16 wherein the one or more action types are selected from the group including: active links in a message, inactive links in a message, electronic mail, phone, auto-correct and relax.

18. The notification method of claim 1 wherein sending the notification comprises:

sending at least one of an electronic mail notification, an active link notification embedded in a message and an inactive link notification embedded in a message.

19. The notification method of claim 18 wherein sending the notification comprises sending the notification in accordance with the notification instruction.

20. The notification method of claim 1 wherein sending the notification comprises:

sending at least one of a telephone notification, an instant messaging notification, a facsimile notification and a page.

21. The notification method of claim 1 further comprising:

receiving an advertiser action instruction in response to the notification; and

automatically adjusting at least one of a cost per click and display rank for the search listing according to the advertiser action instruction, the display rank for the listing defining position of the search listing among search results when the search listing is referred to a

searcher in response to a search query from the searcher, the cost per click being an economic value payable by the owner when the search listing is referred to a searcher in response to a search query from the searcher.

22. The notification method of claim 1 further comprising:

receiving an advertiser action instruction in response to the notification; and

automatically replenishing a balance of an account associated with the owner.

23. The notification method of claim 1 further comprising:

receiving an advertiser action instruction in response to the notification; and

automatically relaxing one or more constraints created by the conditions specified in the notification instruction.

24. The notification method of claim 1 further comprising:

automatically depositing funds in an account associated with the advertiser in response to an account balance too low condition.

25. The notification method of claim 1 further comprising:

automatically adjusting at least one of a cost per click and display rank for the search listing, the display rank for the listing defining position of the search listing among search results when the search listing is referred to a searcher in response to a search query from the searcher, the bid amount being an economic value payable by the owner when the search listing is referred to a searcher in response to a search query from the searcher.

26. A database search system comprising a database of search listings associated with advertisers and a processing system which sends a notification to an advertiser when a change condition of a search listing of the advertiser has occurred.

27. A database search system comprising:

a database of search listings, each search listing being associated with an advertiser;

a search engine; and

means responsive to condition specifying information from one or more advertisers for providing an indication to an advertiser when a specified condition of one or more search listings is satisfied.

28. A database search system comprising:

a database of search listings, each search listing being associated with a respective advertiser, each search listing including a search term and at least one of a variable cost per click (CPC) and a variable display rank;

a search engine configured to identify search listings matching a search query received from a searcher, the matching search listings being ordered in a search result list according to the at least one of the display rank and the bid amount of the matching search listings; and

an agent responsive to a condition definition from an advertiser to provide condition update information to the advertiser.

29. The database search system of claim 28 wherein the agent is configured to receive as the condition definition an indication of search listings and indication of CPC range, and wherein the agent is configured to provide as the condition update information a notification that CPC for the indicated search listings has reached the indicated CPC range.

30. The database search system of claim 28 wherein the agent is configured to receive as the condition definition an indication of search listings and indication of desired rank, and wherein the agent is configured to provide as the condition update information a notification that display rank for the indicated search listings has reached the indicated desired rank.

31. The database search system of claim 28 further comprising an advertiser account management device and wherein the agent is configured to receive as the condition definition an indication of a minimum account balance.

32. The database search system of claim 28 further comprising an advertiser account management device configured to count impressions for specified search listings and wherein the agent is configured to receive as the condition definition an indication of impression-counted search listings and an associated impression limit.

33. The database search system of claim 28 further comprising an advertiser account management device configured to count clicks for specified search listings and wherein the agent is configured to receive as the condition definition an indication of click-counted search listings and an associated click limit.

34. The database search system of claim 28 further comprising an advertiser account management device configured to measure a clickthrough rate for specified search listings and wherein the agent is configured to receive as the condition definition an indication of clickthrough rate search listings and an associated clickthrough rate limit.

35. The database search system of claim 34 wherein the associated clickthrough rate comprises an aggregate clickthrough rate for a combination of the clickthrough rate search listings.

36. The database search system of claim 28 further comprising an advertiser account management device configured to measure an average cost per clickthrough for specified search listings and wherein the agent is configured to receive as the condition definition an indication of average cost per clickthrough search listings and an associated average cost per clickthrough limit.

37. The database search system of claim 28 wherein the agent is configured to receive as the condition definition an indication of the minimum CPC required to attain a given display rank for a search term.

38. A method for operating a database search system, the method comprising:

storing a plurality of search listings in a database, each search listing being associated with an advertiser who gives economic value when a search listing is referred to a searcher;

determining a display position for associated search listings; and

receiving from an advertiser an indication of search listings for which the advertiser desires a notification of a display position change.

39. The method of claim 38 further comprising:

receiving from two or more advertisers positioning information for search listings associated with the two or more advertisers; and

in response to the positioning information, determining the display position.

40. The method of claim 39 wherein receiving the positioning information comprises:

receiving at least one of one of a cost per click and a desired rank for the associated search listings.

* * * * *

APPENDIX B-11



US006108637A

United States Patent [19] Blumenau

[11] **Patent Number:** **6,108,637**
[45] **Date of Patent:** **Aug. 22, 2000**

- [54] **CONTENT DISPLAY MONITOR**
- [75] Inventor: **Trevor Blumenau**, Milpitas, Calif.
- [73] Assignees: **Nielsen Media Research, Inc.**, New York, N.Y.; **Engage Technologies, Inc.**, Andover, Mass.
- [21] Appl. No.: **08/707,279**
- [22] Filed: **Sep. 3, 1996**
- [51] **Int. Cl.**⁷ **G06F 11/30**
- [52] **U.S. Cl.** **705/7; 705/30; 707/10; 707/526**
- [58] **Field of Search** 705/1, 7, 8, 9, 705/30; 706/11; 707/10, 200, 202, 512, 204, 526; 395/200.54, 200.53, 200.59, 712, 182.04, 675, 200.79, 200.47

- [56] **References Cited**
- U.S. PATENT DOCUMENTS**
- 4,283,709 8/1981 Lucero et al. .
- 5,414,809 5/1995 Hogan et al. 345/349
- 5,623,652 4/1997 Vora et al. 707/10
- 5,634,100 5/1997 Capps .
- 5,673,382 9/1997 Cannon et al. 395/182.04
- 5,799,292 8/1998 Hekmatpour 707/513

- FOREIGN PATENT DOCUMENTS**
- 2250112 5/1992 United Kingdom .

- OTHER PUBLICATIONS**
- Brown, "Using Netscape 2—Special Edition", QUE, 2nd edition (attached pages), 1995.
- T. Kamba, "Personalized Online Newspaper", NEC Technical Journal, Jul. 1996, NEC, Japan, vol. 49, No. 7, Abstract.
- Bart Zeigler, "NetCount Seeks to Tally Users of Web Ads," *Wall Street Journal*, Oct. 11, 1996.

Thomas E. Weber, "New Software Helps Advertisers Get Through Tangled Web Pages" *Wall Street Journal*, Oct. 23, 1996.

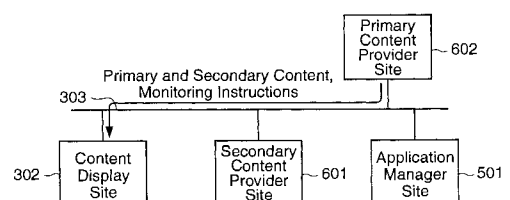
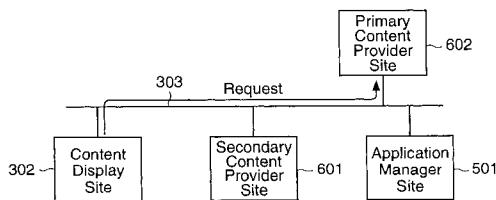
"I/PRO is First to Develop a Solution for Measuring Java Applets," news release from I/PRO World Wide Web site.

Primary Examiner—James P. Trammell
Assistant Examiner—Cuong H. Nguyen
Attorney, Agent, or Firm—David R. Graham

[57] **ABSTRACT**

The invention can enable monitoring of the display of content by a computer system. Moreover, the invention can enable monitoring of the displayed content to produce monitoring information from which conclusions may be deduced regarding the observation of the displayed content by an observer. The invention can also enable monitoring of the display at a content display site of content that is provided by a content provider site over a network to the content display site. Additionally, the invention can enable the expeditious provision of updated and/or tailored content over a network from a content provider site to a content display site so that the content provider's current and appropriately tailored content is always displayed at the content display site. Aspects of the invention related to transfer of content over a network are generally applicable to any type of network. However, it is contemplated that the invention can be particularly useful with a computer network, including private computer networks (e.g., America Online™) and public computer networks (e.g., the Internet). In particular, the invention can be advantageously used with computer networks or portions of computer networks over which video and/or audio content are transferred from one network site to another network site for observation, such as the World Wide Web portion of the Internet.

65 Claims, 7 Drawing Sheets



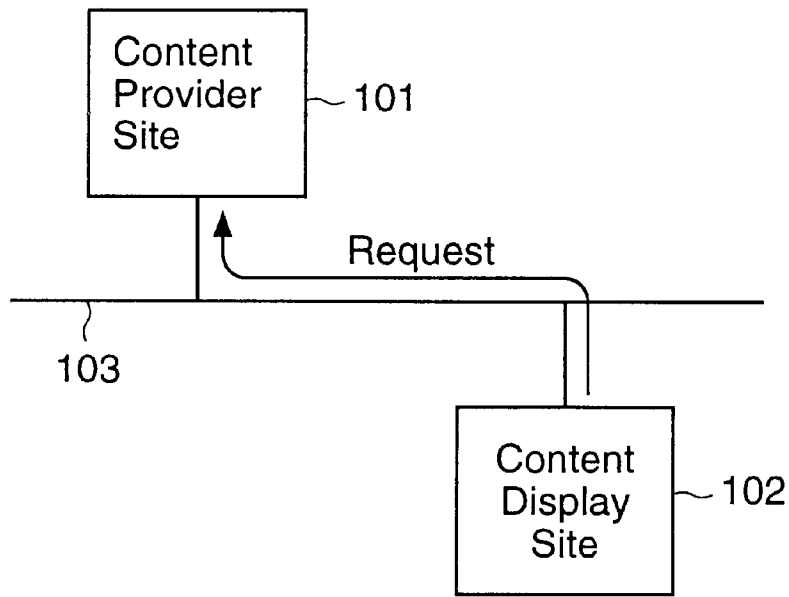


FIG. 1A

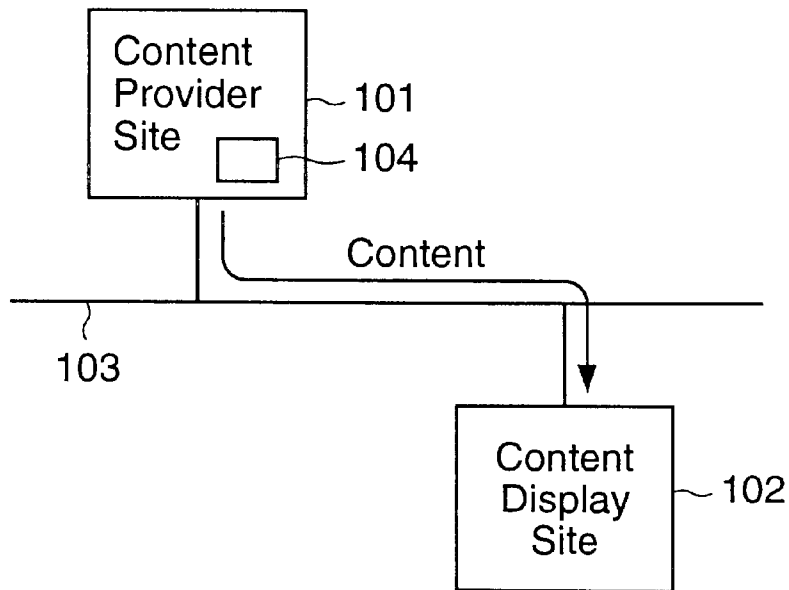
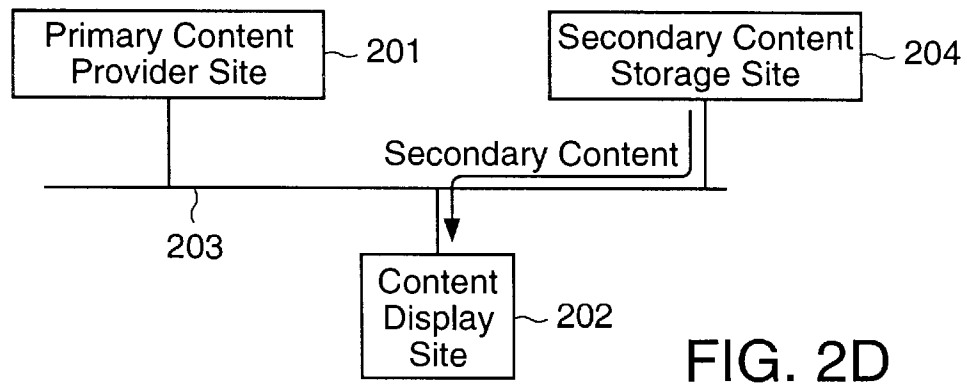
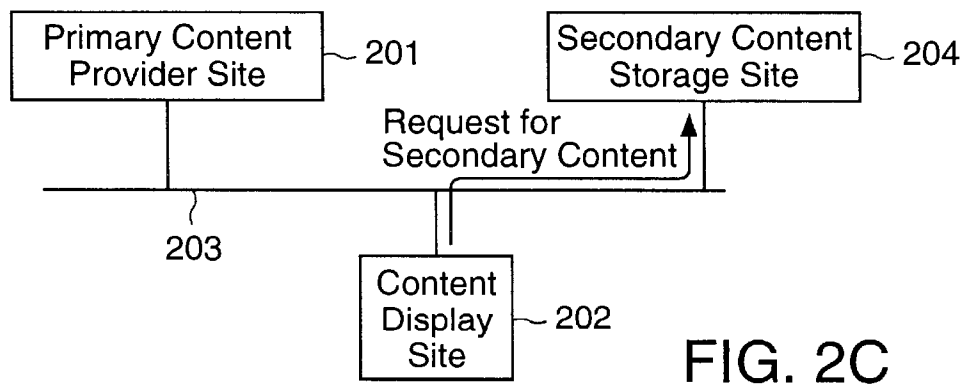
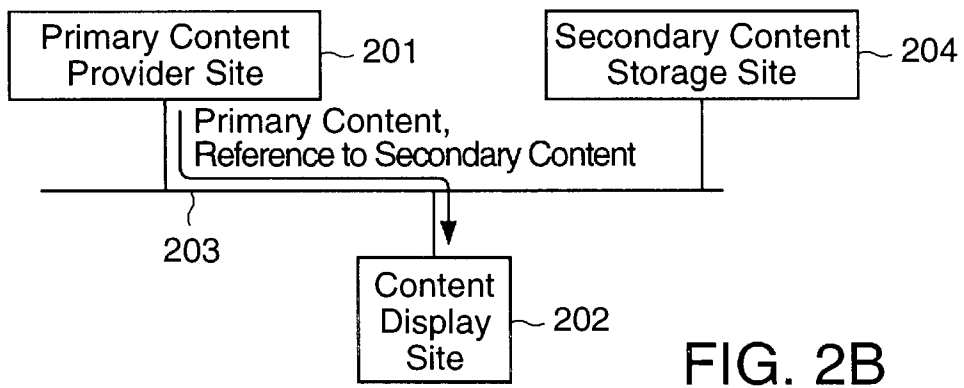
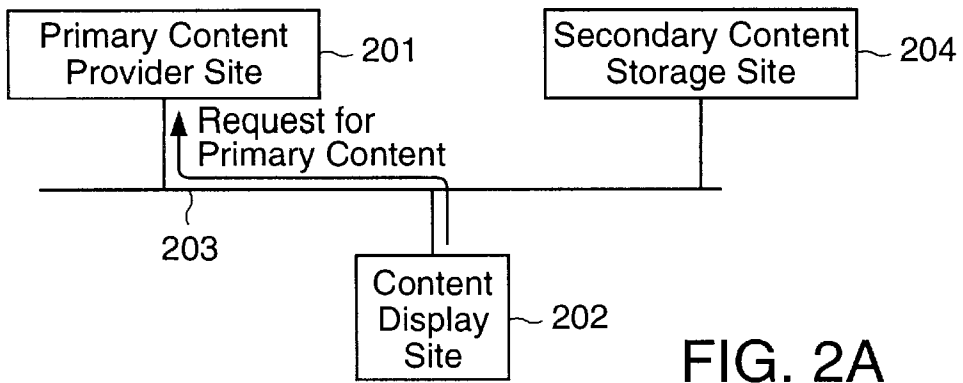


FIG. 1B



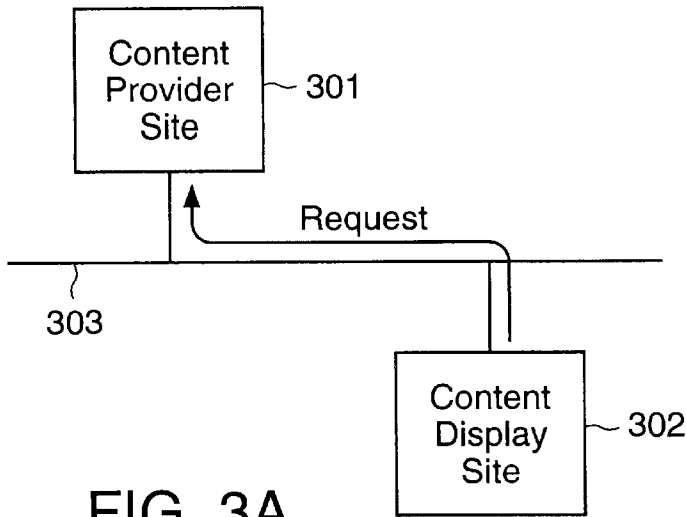


FIG. 3A

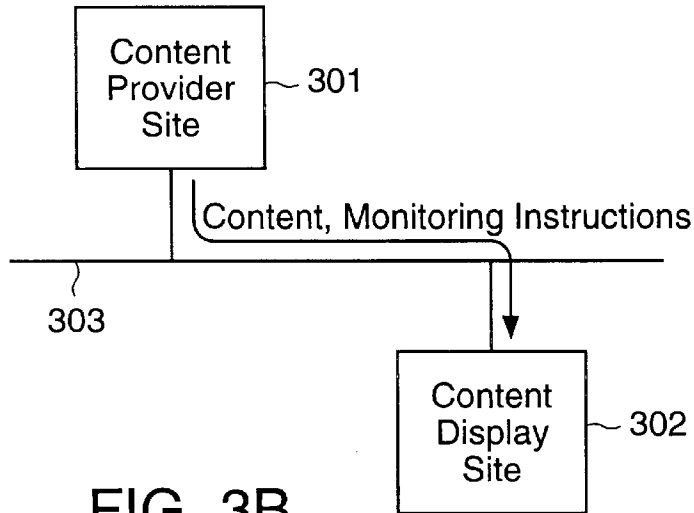


FIG. 3B

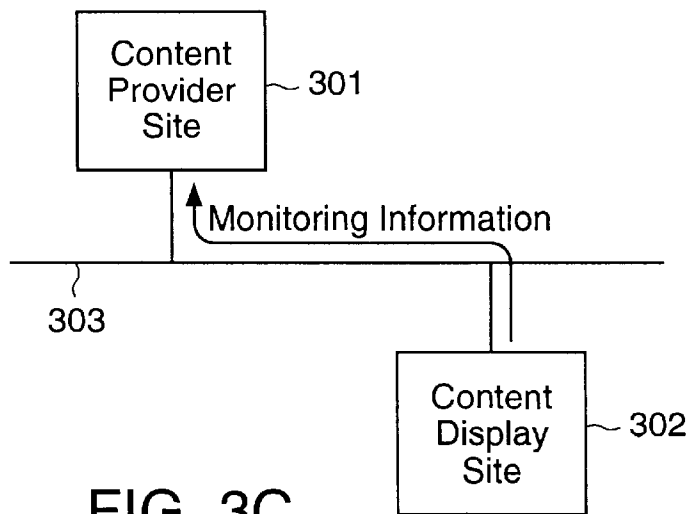


FIG. 3C

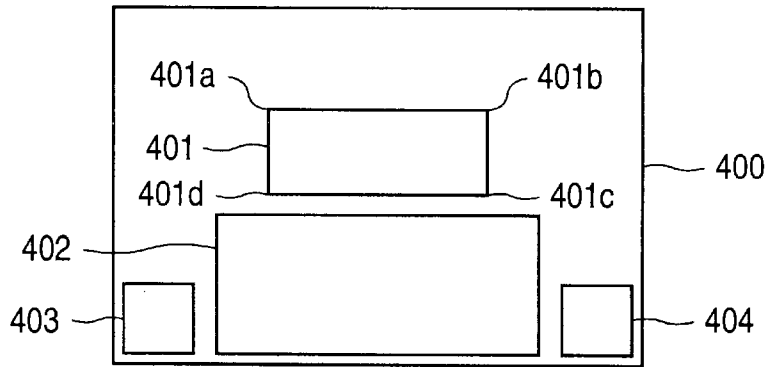


FIG. 4A

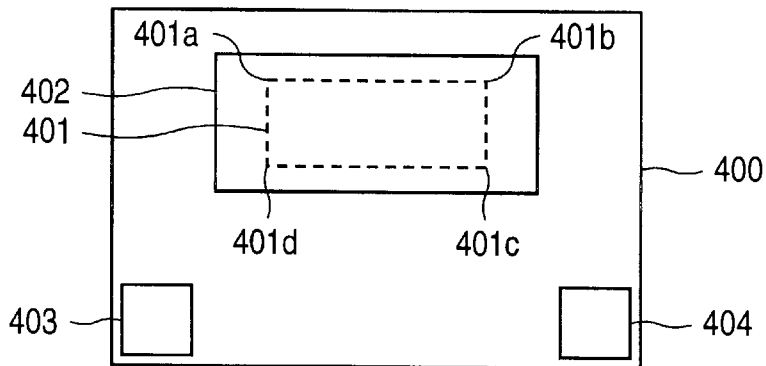


FIG. 4B

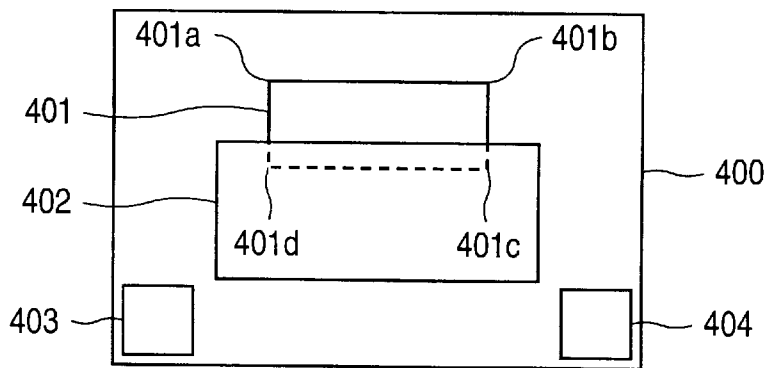


FIG. 4C

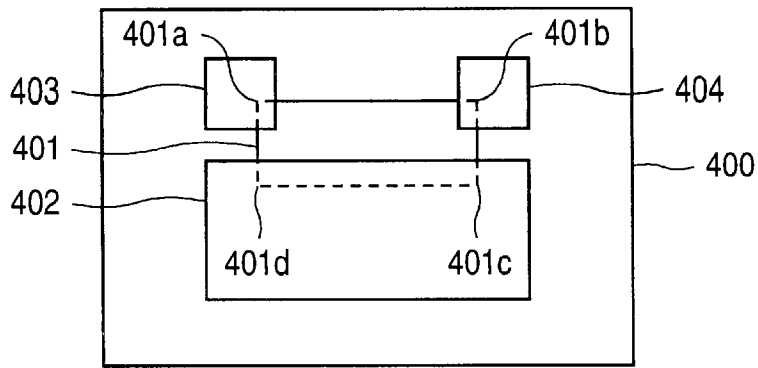


FIG. 4D

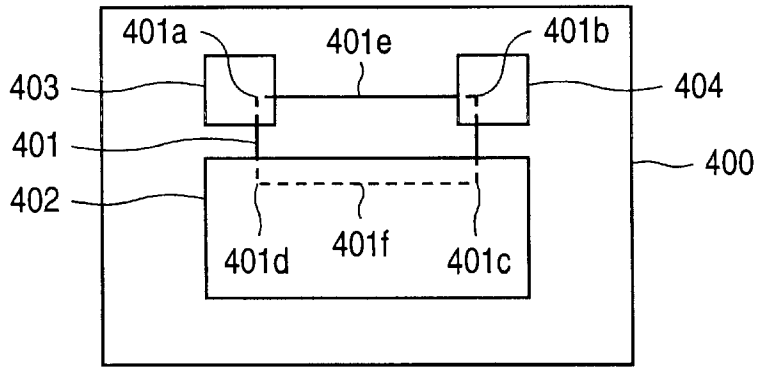


FIG. 4E

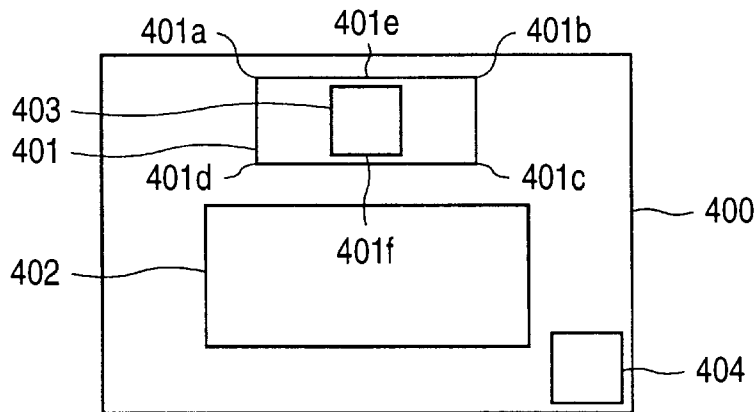


FIG. 4F

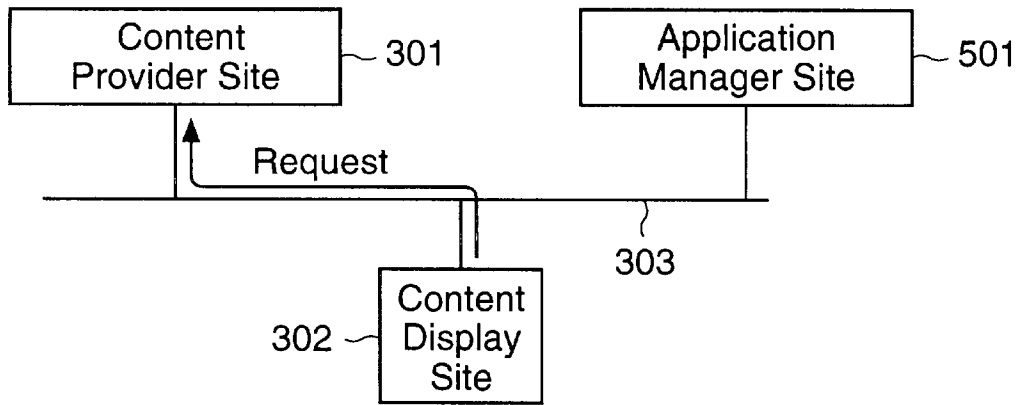


FIG. 5A

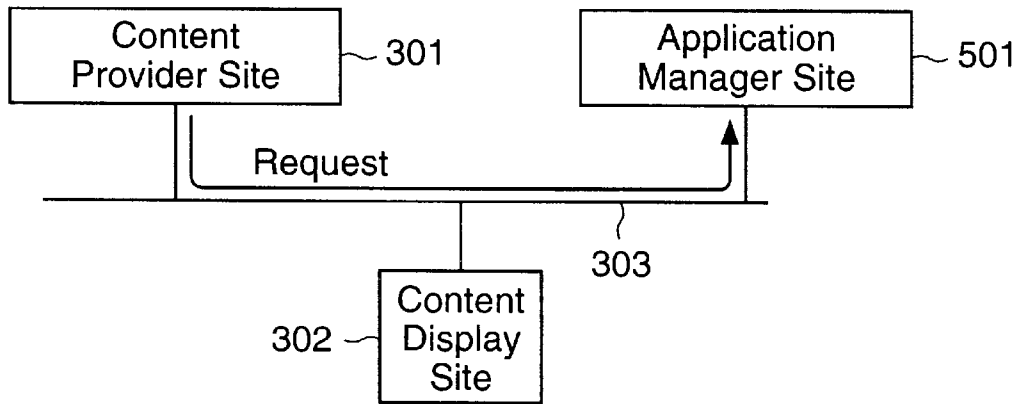


FIG. 5B

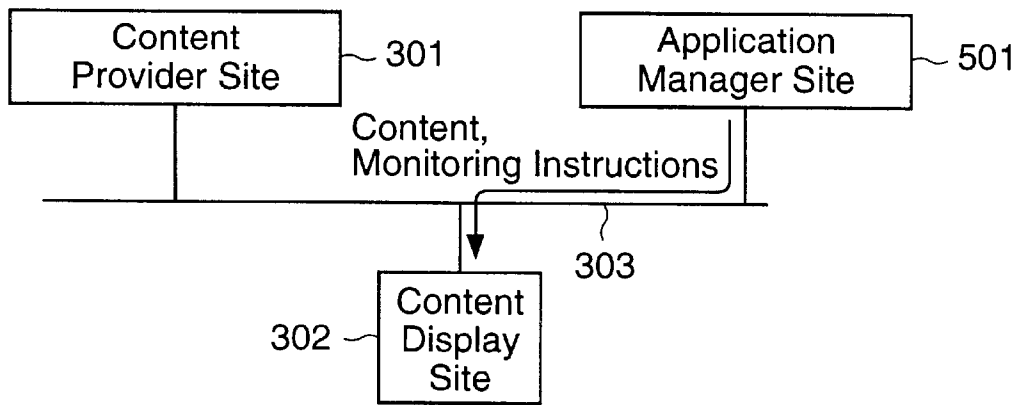


FIG. 5C

FIG. 6A

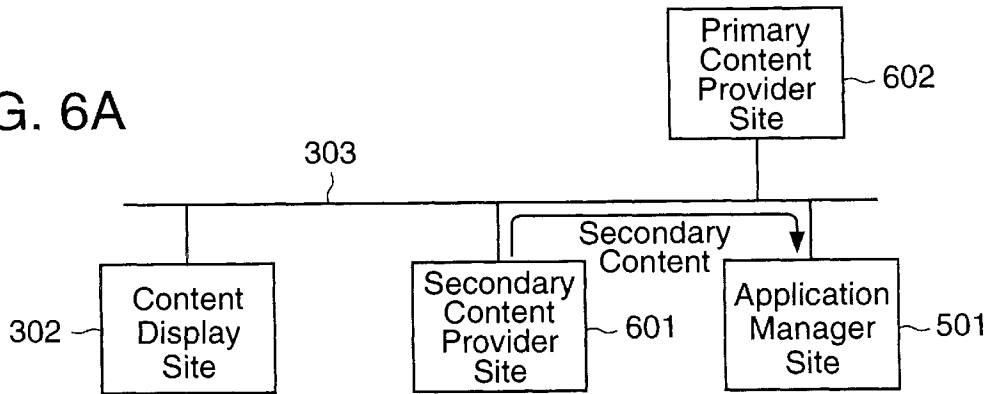


FIG. 6B

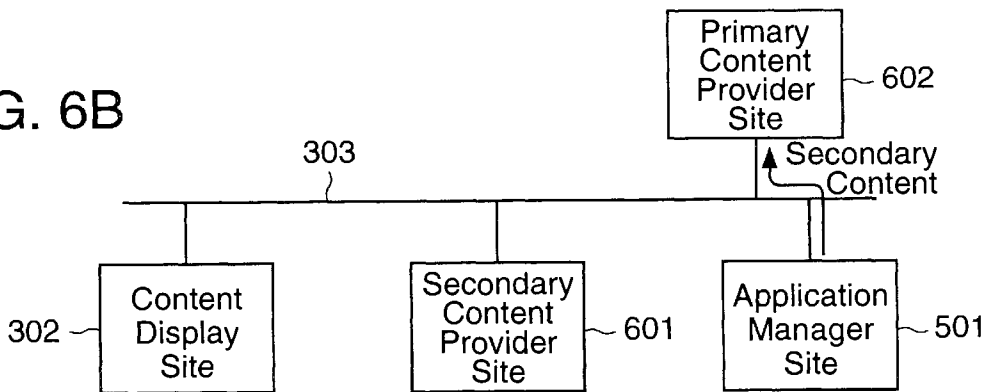


FIG. 6C

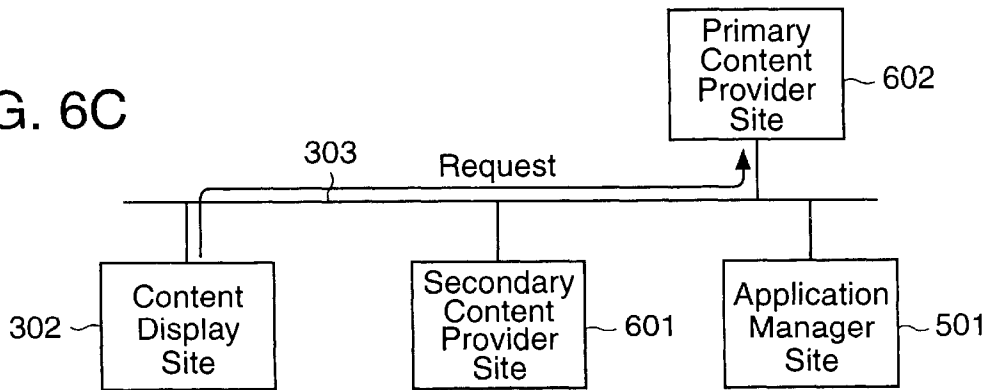
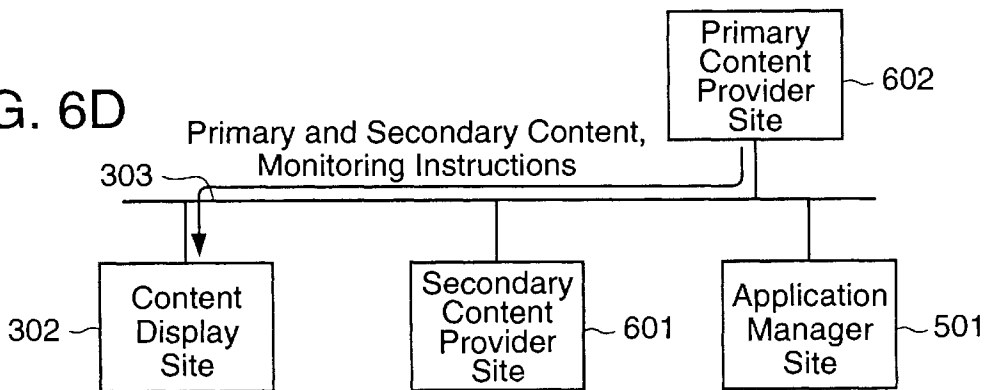


FIG. 6D



CONTENT DISPLAY MONITOR

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to monitoring the display of content by a computer system and observation of that content. The invention also relates to monitoring the display and observation at a content display site of content that is provided by a content provider site over a network to the content display site. The invention further relates to the provision of updated and/or tailored content from a content provider site to a content display site so that the content provider's current content is always displayed at the content display site.

2. Related Art

A large amount of human activity consists of the dissemination of information by information providers (content providers) to information consumers (observers). Recently, computer networks have become a very popular mechanism for accomplishing information dissemination. The use of computer networks for information dissemination has necessitated or enabled new techniques to accomplish particular functions related to the dissemination of information.

For example, information providers of all types have an interest in knowing the extent and nature of observation of the information that they disseminate. Information providers that disseminate information over computer networks also have this interest. However, the use of networked computers for information dissemination can make it difficult to ascertain who is observing the disseminated information and how, since information can be accessed rapidly from a remote location by any of a large number of possible observers whose identity is often not predictable beforehand, and since control over the display of the information once disseminated may not be possible, practical or desirable.

Among information providers, advertisers have particular interest in knowing how and to what extent their advertisements are displayed and/or observed, since such knowledge can be a key element in evaluating the effectiveness of their advertising and can also be the basis for payment for advertising. Mechanisms for obtaining such information have been developed for advertisements disseminated in conventional media, e.g., audiovisual media such as television and radio, and print media such as magazines and newspapers. For example, the well-known Nielsen television ratings enable advertisers to gauge the number of people that likely watched advertisements during a particular television program. As advertising over a computer network becomes more common, the importance of developing mechanisms for enabling advertisers to monitor the display and observation of their advertisements disseminated over a computer network increases.

Previous efforts to monitor the display of advertising (or other content) disseminated over a computer network have been inadequate for a variety of reasons, including the limited scope of the monitoring information obtained, the ambiguous nature of the monitoring information, the incompleteness of the monitoring information, and the susceptibility of the monitoring information to manipulation. Review of some of the techniques that have previously been used to acquire monitoring information regarding the display of content (e.g., advertising) disseminated over a particular computer network—the World Wide Web portion of the Internet computer network—will illustrate the deficiencies of existing techniques for monitoring the display of content disseminated over a computer network.

FIGS. 1A and 1B are simplified diagrams of a network illustrating operation of a previous system for monitoring

requests for content over the World Wide Web. In FIGS. 1A and 1B, a content provider site **101** (which can be embodied by, for example, a server computer) can communicate with a content display site **102** (which can be embodied by, for example, a client computer) over the network communication line **103**. The server computer at the content provider site **101** can store content colloquially referred to as a “Web page.” The client computer at the content display site **102** executes a software program, called a browser, that enables selection and display of a variety of Web pages stored at different content provider sites. When an observer at the content display site **102** wishes to view a particular Web page, the observer causes the client computer at the content display site **102** to send a request to the appropriate server computer, e.g., the server computer at the content provider site **101**, as shown in FIG. 1A. The server computers at content provider sites all include a software program (in the current implementation of the World Wide Web, this is an http daemon) that watches for such incoming communications. Upon receipt of the request, the server computer at the content provider site **101** transfers a file representing the Web page (which, in the current implementation of the World Wide Web, is an html file) to the client computer at the content display site **102**, as shown in FIG. 1B. This file can itself reference other files (that may be stored on the server computer at the content provider site **101** and/or on other server computers) that are also transferred to the content display site **102**. The browser can use the transferred files to generate a display of the Web page on the client computer at the content display site **102**. The http daemon, in addition to initiating the transfer of the appropriate file or files to the content display site **102**, also makes a record of requests for files from the server computer on which the daemon resides. The record of such requests is stored on the server computer at the content provider site **101** in a file **104** that is often referred to as a “log file.”

The exact structure and content of log files can vary somewhat from server computer to server computer. However, generally, log files include a list of transactions that each represent a single file request. Each transaction includes multiple fields, each of which are used to store a predefined type of information about the file request. One of the fields can be used to store an identification of the file requested. Additional fields can be used to store the IP (Internet Protocol) address of the client computer that requested the particular file, the type of browser that requested the file, a time stamp for the request (i.e., the date and time that the request was received by the server computer), the amount of time required to transfer the requested file to the client computer, and the size of the file transferred. Other information about file requests can also be stored in a log file.

Previous methods for monitoring the display of content distributed over the World Wide Web have used the information stored in the log file. For example, one previous method has consisted of simply determining the number of transactions in the log file and counting each as a “hit” on a Web page, i.e., a request for a Web page. The number of hits is deemed to approximate the number of times that the Web page has been viewed and, therefore, the degree of exposure of the content of the Web page to information consumers.

There are a number of problems with this approach however. For example, as indicated above, a request for a Web page may cause, in addition to the request for an initial html file, requests for other files that are necessary to generate the Web page. If these other files reside on the same server computer as the initial html file, additional transac-

tions are recorded in the log file. Thus, a request for a single Web page can cause multiple transactions to be recorded in the log file. As can be appreciated, then, the number of times that a Web page is transferred to a content display site can be far less than the number of transactions recorded in the log file. Moreover, without further analysis, there is no way to accurately predict the relationship between the number of transactions in the log file and the number of times that a Web page has been transferred to the content display site. Such inaccuracy can be very important to, for example, advertisers—whose cost of advertising is often proportional to the measured exposure of the advertising—since the measured exposure of their advertising (and, thus, its cost) may be based upon the number of hits on a Web page containing their advertisement.

A method to overcome this problem has been used. By analyzing the contents of the log file to determine which file was requested in each transaction, it may be possible to differentiate transactions in which the initial html file needed to generate a Web page is requested from transactions in which the requested file is one which is itself requested by another file, thus enabling “redundant” transactions to be identified and eliminated from the hit count. While such an approach can increase the accuracy of counting Web page hits, it still suffers from several problems.

For example, log file analysis may result in some undercounting of Web page hits, apart from any overcounting. This is because, once transferred to a client computer at a content display site, the files necessary to generate a Web page can be stored (“cached”) on that client computer, thus enabling an observer at the content display site to view the Web page again without causing the client computer to make another request to the content provider server computer from which the Web page was initially retrieved. Consequently, the observer can view the Web page without causing transactions to be added to the log file, resulting in undercounting of the number of Web page hits.

Additionally, log files are subject to manipulation, either directly or indirectly. For example, an unscrupulous content provider could directly manipulate the log file by retrieving and editing the log file to add phony transactions, thus artificially increasing the number of Web page hits and making the Web page appear to be more popular than it really is. This problem can be ameliorated by causing the log files to be transferred periodically at predetermined times (e.g., each night at 12:00 midnight) from the server computer at the content provider site to a neutral network site; however, the log file can still be manipulated during the time between transfers.

A log file might be manipulated indirectly, for example, by programming one or more computers to continually request a Web page, thereby generating a large number of hits on that Web page. While the log file would contain transactions corresponding to actual file requests associated with the Web page, these requests would be artificial requests that would almost certainly not result in a display of the Web page, and certainly not in the observation of the Web page. Moreover, checking the contents of the log file for an unusually high number of requests from a particular IP address (i.e., client computer) may not enable such manipulation to be detected, since a large number of requests may legitimately come from a client computer that serves many users (for example, the proprietary network America Online™ has a handful of computers that are used by many users of that network to make connection to the Internet and World Wide Web).

It may be possible to identify the real origin of requests for content using “cookies.” A cookie enables assignment of

a unique identifier to each computer from which requests really emanate by transferring the identifier to that computer with content transferred to that computer. Future requests for content carry this identifier with them. The identifier can be used, in particular, to aid in identification of indirect log file manipulation, as described above, and, more generally, to enable more robust log file analysis.

Notwithstanding such enhancement, cookies do not overcome a fundamental problem with the use and analysis of log files to ascertain information regarding the display of content provided over the World Wide Web. That is, as highlighted by the overcounting problem associated with the above-described artifice and the undercounting problem associated with caching of content at the content display site, log files only store information about file requests. A log file does not even indicate whether the requested file was actually transferred to the requesting client computer (though, typically, such file transfer would occur). Nor does a log file include any information about how the file was used once transferred to the requesting client computer. In particular, log files do not provide any information regarding whether the content represented by the requested file is actually displayed by the client computer at the content display site, much less information from which conclusions can be deduced regarding whether—and if so, how—the content was observed by an observer. These limitations associated with the content of a log file cannot be overcome by a monitoring approach based on log file analysis. Moreover, log file analysis is calculation intensive, requiring hours in some instances to extract the desired information from the log file.

Another method of monitoring the display of content disseminated over the World Wide Web uses an approach similar to that of the Nielsen ratings system used in monitoring television viewing. In this method, monitoring software is added to the browser implemented on the client computers of a selected number of defined observers (e.g., families) to enable acquisition of data regarding advertising exposure on those computers. This information is then used to project patterns over the general population.

However, this approach also has several disadvantages. First, only a limited amount of data is collected, i.e., data is only obtained regarding a small number of information consumers. As with any polling method, there is no guarantee that the data acquired can be extrapolated to the general population, even if the observers selected for monitoring are chosen carefully and according to accepted sampling practices. Second, as the size of the World Wide Web (or other computer network for which this method is used) grows, i.e., as the number of content provider sites increases, the number of monitored observers necessary to ensure accurate representation of the usage of all content provider sites must increase, since otherwise there may be few or no observer interactions with some content provider sites upon which to base projections. It may not be possible to find an adequate number of appropriate observers to participate in the monitoring process, particularly given concerns with the attendant intrusion into the privacy of the selected observers. Third, installation of the monitoring software on a client computer to be compatible with a browser presents a number of problems. Such installation requires active participation by observers; since observers typically do not reap benefit from operation of the monitoring software, they may be reluctant to expend the effort to effect installation. The monitoring software must continually be revised to be compatible with new browsers and new versions of old browsers. To enable monitoring of a large number of client

computers, the software must be tested for compatibility with a wide variety of computing environments. And, as currently implemented, such monitoring software is also dependent upon the computing platform used, making it necessary to revise the monitoring software for use with new computing platforms or risk skewing the demographics of the sample users.

In addition to desiring information regarding the display and observation of the content that they provide, content providers also often desire to provide content to a content display site that is particularly tailored for observation (e.g., according to various demographic characteristics of an expected observer) at that content display site. For example, text content should be expressed in a language that the observer can understand. If appropriate for the content, it is desirable to tailor the content according to, for example, the age, sex or occupation of the observer.

Such tailoring of content has previously been enabled by modifying the http daemon on a computer at the content provider site to cause a particular version of a set of content to be transferred to a requesting content display site based upon the IP address of that content display site. While such tailoring of content is useful, it is desirable to be able to tailor the presentation of content in additional ways not enabled by this approach.

Content providers also often desire to provide their content with the content of other content providers. For example, it is a common practice for content providers (referred to here as “primary content providers”) on the World Wide Web to include advertisements from other entities (referred to here as “secondary content providers”) as part of the content provider’s Web page. In such situations, it is desirable for the secondary content provider to be able to easily update and/or appropriately tailor (e.g., according to characteristics of the requester) the content that they supply to the primary content provider. This could be accomplished by causing the primary content provider site to contact the secondary content provider site—each time that the primary content provider receives a request for content that includes the secondary content—to retrieve the secondary content (thus ensuring that updated, appropriately tailored secondary content is used) or check whether updated or tailored secondary content is available (if so, the content is retrieved). (This method could also be modified so that content retrieval or a check for updated and/or tailored content is only performed according to a predetermined schedule.) However, both the primary content provider and the secondary content provider may not want their systems burdened with the extra computational capacity required to handle the multitude of requests that would be needed to effect this operation. Alternatively, the primary content provider could collect and store the updated and tailored content from the secondary content providers at the primary content provider site. However, the burden associated with collecting and managing the content from secondary content providers may be more than the primary content provider wants to shoulder.

One way that this functionality can be achieved without creating an undesirable burden on the primary or secondary content providing systems is by providing a secondary content storage site that can continually store the most recent content provided by a secondary content provider, as well as different sets of content tailored for particular situations (e.g., display by particular observers or at particular times). FIGS. 2A through 2D are simplified diagrams of a network illustrating the operation of such a system. In FIG. 2A, a content display site 202 makes a request over the network

communication line 203 to the primary content provider site 201 for content that includes the secondary content. In FIG. 2B, the primary content provider site 201 transfers the file or files stored at the primary content provider site 201 that are necessary to generate a display of the primary content. These files include appropriate reference to a file or files stored at a secondary content storage site 204 that includes the most updated and/or appropriately tailored secondary content for display with the primary content. As shown in FIG. 2C, this reference causes the content display site 202 to request the secondary content from the secondary content storage site 204. In FIG. 2D, the secondary content is transferred from the secondary content storage site 204 to the content display site 202 for display at the content display site 202.

However, while this system can relieve the primary content provider of the burden of managing the acquisition, storage and provision of secondary content (a burden that can become rather onerous when many secondary content providers are providing content to the primary content provider), the system has a characteristic that can make it undesirable for many content providers. The secondary content storage site not only manages the secondary content, it also provides the secondary content when requests for primary content are made to the primary content provider. Moreover, the secondary content is frequently content, such as graphics files used to generate visual images (which frequently dominate advertisements), that has a high bandwidth requirement for transmission over the network. By taking control of the transmission of secondary content to the content display site, the secondary content storage site is also frequently taking control of the most bandwidth sensitive parts of the content provided by the primary content providers. The operator of the secondary content storage site may not provide a system that addresses the bandwidth requirements to the satisfaction of the primary content provider, so that the presentation of the combined primary and secondary content occurs more slowly than desired by the primary content provider. Thus, this approach causes the primary content provider to lose control of a critical aspect of their operation.

SUMMARY OF THE INVENTION

The invention can enable monitoring of the display of content by a computer system. Moreover, the invention can enable monitoring of the content display to produce monitoring information from which conclusions may be deduced regarding the observation of the content display by an observer. The invention can also enable monitoring of the display at a content display site of content that is provided by a content provider site over a network to the content display site. Additionally, the invention can enable the expeditious provision of updated and/or tailored content over a network from a content provider site to a content display site so that the content provider’s current and appropriately tailored content is always displayed at the content display site.

Aspects of the invention related to transfer of content over a network are generally applicable to any type of network. However, it is contemplated that the invention can be particularly useful with a computer network, including private computer networks (e.g., America Online™) and public computer networks (e.g., the Internet). In particular, the invention can be advantageously used with computer networks or portions of computer networks over which video and/or audio content are transferred from one network site to another network site for observation, such as the World Wide Web portion of the Internet. Additionally, the invention is

particularly useful in monitoring the display of content obtained over such a network using an interactive browser to acquire and view the content in real time.

In one aspect of the invention, the display of content by a computer system can be monitored by monitoring the position of the content display on a display screen of the computer system and evaluating the position of the content display on the display screen to produce monitoring information regarding display of the content. Monitoring of content display according to this aspect of the invention can be further enabled by monitoring the position of one or more other images on the display screen and comparing the position of the content display to the position of the other images to produce the monitoring information. In particular, this aspect of the invention can enable a determination as to whether (and for how long) the content display is hidden by one of the other images, and, further, whether the content display is fully hidden or partially hidden (and for how long the content display is fully and partially hidden, respectively). This information can be useful to, for example, indicate the amount of time that the content display was visible to an observer for observation, or to aid the content provider in determining in which regions of a display screen his content is most likely to be unobstructed. This aspect of the invention can also enable determination of the number of times that an on-screen pointer (e.g., a mouse arrow or a cursor) entered an area defined by the content display. This information may be useful in determining how attentive the observer was to the content, since an observer frequently watches the position of the on-screen pointer when viewing the display screen.

In another aspect of the invention, the display of content by a computer system can be monitored by monitoring the change in time of a characteristic of the content display and evaluating the change in time of the characteristic of the content display to produce monitoring information regarding display of the content. Monitoring of content display according to this aspect of the invention can be further enabled by monitoring the change in time of a characteristic of the computer system and comparing the change in time of the characteristic of the content display to the change in time of the characteristic of the computer system to produce the monitoring information. This aspect of the invention can also enable, as discussed above, determination as to whether (and for how long) the content display is fully or partially hidden by another displayed image, as well as the number of times that an on-screen pointer entered an area defined by the content display.

In still another aspect of the invention, in a computer system in which the content is displayed in response to an instruction that is provided from a source external to the computer system and the system for monitoring (e.g., an instruction provided by a user of the computer system), the beginning and end of a display of the content can be ascertained so that monitoring of the display of content by the computer system can begin at the beginning of the content display and end at the end of the content display. The monitoring can occur in accordance with other aspects of the invention described herein. The monitoring can also determine the duration of the display of the content. Since the occurrence of monitoring according to this aspect of the invention is coincident with the display of the content to be monitored, the monitoring expends processing capability of the computer system only when necessary, while simultaneously assuring that monitoring occurs at all times that the content is displayed.

In yet another aspect of the invention, where content is provided by a content provider site over a network to a

content display site for display at the content display site, a mechanism for monitoring the display of the content can be transferred from the content provider site to the content display site in response to (e.g., together with) the transfer of content from the content provider site. The monitoring can occur in accordance with other aspects of the invention described herein. Monitoring information obtained regarding the display of the content at the content display site can be transferred to a remote site that is part of the network. The remote site can, but need not necessarily be, the content provider site from which the content was transferred to the content display site. Where the remote site is such content provider site, the monitoring information can then, in turn, be transferred from the content provider site to a second remote site. Further, where the remote site is such content provider site, the monitoring information can be transferred from the content display site to the content provider site via a communication means that is different from the communication means used to transfer the content from the content provider site to the content display site, a feature that can be useful, for example, when the network is the World Wide Web. This aspect of the invention provides a heretofore unavailable monitoring capability for obtaining information about how content is displayed on a network such as the World Wide Web. In particular, it has not previously been possible to monitor content transferred from a content provider site on the World Wide Web once the content has been transferred to a content display site.

In a still further aspect of the invention, in a network which operates according to a protocol that enables new content to be transferred to a content display site in response to selection of a portion of the content currently being displayed at the content display site, a mechanism for monitoring the display of content can be transferred from a content provider site to a content display site so that the mechanism for monitoring operates at the content provider site. The monitoring can occur in accordance with other aspects of the invention described herein. Monitoring information obtained regarding the display of the content at the content display site can be transferred to a remote site that is part of the network. This aspect of the invention is particularly advantageous when at least some of the content being monitored comprises a graphical display. As discussed above with respect to the immediately preceding aspect of the invention, this aspect of the invention provides a heretofore unavailable monitoring capability for obtaining information about how content is displayed when retrieved over a network in an interactive browsing environment, such as occurs on the World Wide Web.

In yet a further aspect of the invention, the display of content that is provided by a content provider site over a network to the content display site is monitored to produce monitoring information, then the monitoring information is transferred to a remote site of the network that is different from the content provider site. According to this aspect, the monitoring information can first be transferred to the content providing site before eventual transfer to the remote site, so long as the monitoring information cannot be stored at the content provider site, or accessed or manipulated at the content provider site before transfer to the remote site. Access to the monitoring information at the remote site can be allowed to enable interaction with, but not modification of, the monitoring information. This aspect of the invention provides a system configuration that can overcome the problem of possible tampering with the substance of the monitoring information by the content provider. Further, this aspect of the invention can be implemented so that the

content and the monitoring instructions are stored at the remote site and transferred to the content display site when requested by an observer at the content display site, thus relieving the content provider of storing and managing the content and monitoring instructions at the content provider site and thereby simplifying use of the invention for the content provider.

In another aspect of the invention, current and/or tailored content can be provided to a content display site from a content provider site. The content can include both primary content (from the content provider site) and secondary content (provided by third parties). The primary and secondary content can be provided from a secondary content provider site to an application manager site. When the application manager site receives new content (in particular, updated and/or tailored content) from any content provider site, that content is transferred to content provider sites that use that content. Updated and/or tailored content is therefore available for transfer to a content display site immediately upon receipt of a request for the content from the content display site. This aspect of the invention relieves the primary content provider of the need to manage the storage of content, while reserving control over the provision of that content to the primary content provider, thereby enabling the content provider to ensure that the bandwidth requirement of the content provided from the content provider site are met.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A and 1B are simplified diagrams of a network illustrating operation of a previous system for monitoring requests for content over the World Wide Web.

FIGS. 2A, 2B, 2C and 2D are simplified diagrams of a network illustrating operation of a previous system for enabling retrieval of updated and/or tailored secondary content for use in primary content provided over the network.

FIGS. 3A, 3B and 3C are simplified diagrams of a network illustrating operation of one embodiment of the invention.

FIGS. 4A, 4B and 4C are simplified views of a display screen including a content display and other images, illustrating an unobstructed, fully hidden, and partially hidden content display, respectively.

FIG. 4D is a simplified view of a display screen including a content display and other images, illustrating a content display that is only partially hidden, but that would be determined to be fully hidden according to a method of the invention.

FIG. 4E is a simplified view of the display screen shown in FIG. 4D, illustrating how another method of the invention can correctly determine the content display to be partially hidden.

FIG. 4F is a simplified view of a display screen including a content display and other images, illustrating a display that is partially hidden, but that may be determined to be unobstructed according to a method of the invention.

FIGS. 5A, 5B and 5C are simplified diagrams of a network illustrating operation of another embodiment of the invention.

FIGS. 6A, 6B, 6C and 6D are simplified diagrams of a network illustrating operation of still another embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention includes several aspects related to the display of content to an observer. For example, the invention

can enable monitoring of the display of content by a computer system. In particular, the invention can enable monitoring of the displayed content in a manner that provides monitoring information from which aspects of the observer's observation of the content can be gleaned. The invention can also enable monitoring of the display and—using the aforementioned capability—the observation at a content display site of content that is provided by a content provider site over a network to the content display site. Additionally, the invention can enable the expeditious provision of updated and/or tailored content over a network from a content provider site to a content display site so that the content provider's current and appropriately tailored content is always displayed at the content display site.

Herein, "content" refers generally to any sensory images (or data used to produce those sensory images) displayed by a device with which the invention is used. "Observation" refers to the perception of content by an observer. Typically, the content will be visual or aural images produced by the device; observation of such content thus consists of viewing or listening, as appropriate, to the produced images.

Certain aspects of the invention relate to the monitoring of content obtained from, or provision of content over, a network. "Content provider site" refers to a device that is part of the network and that can provide content to another device that is part of the network. "Content display site" refers to a device that is part of the network and that can receive and display content from another device that is part of the network. It is contemplated that the invention can be particularly useful with a computer network that operates in this way. "Computer network" includes any collection of interconnected computer systems. "Computer system" refers to a device or collection of devices which depend upon a computational device (e.g., a general or special purpose processor) for at least some aspects of their operation. In particular, as used herein, a "computer system" can include any type of display device, including a conventional computer display monitor, a television, or one or more audio speakers.

FIGS. 3A, 3B and 3C are simplified diagrams of a network illustrating operation of one aspect of the invention. A content display site 302 (which can be embodied by a conventional client computer) is linked via a network communication line (or lines) 303 to a content provider site 301 (which can be embodied by a conventional server computer). (Typically, the network links multiple content display sites with multiple content provider sites; a single content display site 302 and a single content provider site 301 are shown in FIGS. 3A, 3B and 3C for simplicity. Additionally, it is to be understood that each site on the network can function as both a content display site and a content provider site.) As shown in FIG. 3A, the client computer at the content display site 302 requests content from the server computer at the content provider site 301 over the network communication line 303. As shown in FIG. 3B, the server computer at the content provider site 301 provides content to the client computer at the content display site 302 over the network communication line 303. According to this aspect of the invention, in response to the request for content from the content provider site 301, a set of monitoring instructions (which can be embodied, for example, in a computer program) are also transferred to the content display site 302. The transfer of the monitoring instructions can occur before, with or after the transfer of the content. As explained in more detail below, the monitoring instructions cause the client computer at the content display site 302 to monitor the display of the content to produce

monitoring information regarding the manner in which the content is displayed. As shown in FIG. 3C, the monitoring information is transferred from the content display site 302 to the content provider site 301 over the network communication line 303. (The monitoring information could, alternatively or additionally, be transferred to another site that is part of the network.) Review of the monitoring information produced by the monitoring instructions can enable conclusions regarding the observer's observation of the content to be deduced, as explained in more detail below. (It should be noted that, more generally, monitoring instructions according to the invention can be used to monitor the display of content on a computer system whether or not the computer system is part of a network and receives content and monitoring instructions over the network.)

The invention can be used with both public computer networks (e.g., the Internet) and private computer networks (e.g., commercial online services such as America Online™, Prodigy™ and CompuServe™, as well as intranets). In particular, the invention can be advantageously used with computer networks or portions of computer networks over which video and/or audio content are transferred from one network site to another network site for display. Further, the invention can advantageously be used with a network in which the network sites can be accessed in real time with a browser. ("Browser" can refer to a computer program which can interpret hypertext files and display the content corresponding to those files, as well as enable transfer from one hypertext file to another via a hyperlink within the hypertext file being transferred from.) The World Wide Web portion of the Internet is a well-known current example of such a network with which the invention can be used. Below, some aspects of the invention are described, for purposes of illustration, as implemented in a manner that is compatible with the World Wide Web, i.e., in accordance with the hypertext markup language (html) and the hypertext transfer protocol (http). However, none of the aspects of the invention are limited to such implementation.

When the invention is used with a computer network or to monitor display of content by a computer system, aspects of the invention can be implemented as one or more computer programs that can be executed by a computer to achieve the functionality of that aspect. Generally, such computer programs can be implemented using any appropriate computer programming language. However, when an aspect of the invention is used with a computer network that includes computers of many different types (such as the Internet), the computer programming language is preferably one that can be executed by any type of computer (i.e., the computer programming language is platform independent). The Java programming language, developed by Sun Microsystems, Inc. of Mountain View, Calif., is one such computer programming language. Below, some aspects of the invention are described, for purposes of illustration, as implemented in the Java programming language. Again, however, none of the aspects of the invention are limited to such implementation.

In one embodiment of the invention, the monitoring instructions are transferred to the content display site 302 together with the content. In a particular embodiment, the monitoring instructions are part of a computer program that also includes instructions for displaying the content. Illustratively, such a computer program can be an applet written in the Java programming language. As will be appreciated by those skilled in the use of html, Example 1 below illustrates a set of instructions in accordance with the html syntax that can be used to cause execution of an applet that both displays content and monitors the display.

EXAMPLE 1

```

5 <applet code="AdInsert.class" width=230 height=33>
  <param name="image" value="images/southwest.gif">
  <param name="href" value="http://www.swa.com/">
  </applet>

```

The instructions shown in Example 1 are executed by a conventional browser implemented on a computer at a content display site when an observer at the content display site makes a request for (e.g., selects a hyperlink) the content represented by the file "southwest.gif." The request is received by an http daemon at the appropriate content provider site. The instructions identify the location ("image") at the content provider site of an applet (a small application program) called "AdInsert" that includes further instructions which, when executed, perform a monitoring method according to the invention, as well as cause the content to be displayed. (The steps that can be implemented in such a monitoring method are discussed further below.) Upon receipt of the request by the http daemon at the content provider site, the AdInsert applet is transferred to the requesting content display site and begins executing. The instructions in Example 1 also establish the size of the area (width and height) in which the content is displayed on a computer display screen, as well as indicate a network site ("href") to which connection can be made by selecting a hyperlink within the content. Thus, illustratively, in accordance with the invention, content from a content provider site that can be accessed by a browser (such as a network site that is part of the World Wide Web) can be transferred to and displayed at a content display site by transferring an applet to the content display site that can be executed by the browser to both display the content and cause aspects of the display of the content to be monitored. (Note that the content being monitored can comprise all of the content being displayed, or only a part of the content being displayed, e.g., an advertisement present in a Web page.)

In contrast, previously, only content has been transferred to content display sites, using an html syntax as shown in Example 2 below for the content that is displayed by the html syntax shown in Example 1 above.

EXAMPLE 2

```

50 <a href="http://www.swa.com/">
  
  </a>

```

Thus, previously, it has not been possible to monitor content transferred from a content provider site on the World Wide Web once the content has been transferred to the content display site. As can be appreciated, then, this aspect of the invention provides a powerful tool, not previously available, for obtaining information about how content is displayed on a computer network such as the World Wide Web.

Implementation of a monitoring method as described immediately above means that the operation of the monitoring method is coincident with the display of the content to be monitored. Since the monitoring method does not operate when the content is not being displayed, the monitoring method expends processing capability of the computer system at the content display site only when necessary. At the same time, operation of the monitoring method at all times when the content is displayed is assured.

Further, since the monitoring method can be implemented as part of a broader method according to the invention that also causes the content to be displayed, the problems previously noted with monitoring the display of content that is cached at the content display site are overcome. This is because, unlike the previous use of log files—which require that a request be made to a content provider site in order that any information be recorded—a monitoring method according to the invention can record monitoring information any time that the content is displayed, without regard to the manner in which the display is requested. In particular, the invention can enable the number of times that a particular set of content is displayed to be precisely counted. This is a huge improvement over previous methods as described above, which not only do not count the number of times that the content is displayed (they count requests), but may not even count the number of requests accurately.

The instructions that implement a monitoring method according to the invention can be used to obtain a large variety of monitoring information. For example, the contents of conventional log files (discussed above) can be ascertained by a monitoring method of the invention. An important aspect of the invention, however, is that monitoring information beyond that available in a conventional log file can also be obtained by a monitoring method according to the invention. Instructions for obtaining several types of such monitoring information are described below. However, it is to be understood that the descriptions below are merely illustrative of the types of monitoring information that can be obtained; the obtaining of other types of monitoring information is also contemplated by the invention.

For example, a monitoring method according to the invention can detect each time that the content is displayed. In fact, in one embodiment of a monitoring method according to the invention, the monitoring method does no more than this. A monitoring method that detects the display of the content can be implemented by an applet as described above. The “monitoring instructions” of such an applet may be no more than an instruction that causes an indication that the applet has executed to be stored or transferred to an appropriate network site (discussed further below). A monitoring method that can ascertain whether the content was displayed or not can be a very useful monitoring method that provides important basic information not previously available in an interactive browsing environment for acquiring and viewing content. In particular, undercounting (due to, for example, caching of content at the content display site) and overcounting (due to, for example, submission of artificial requests for content that do not result in the display of content) of the number of times that the content is displayed are avoided.

A monitoring method according to the invention can also determine the duration of the content display. For example, the duration of the content display can be determined as the amount of time that the computer program for displaying the content executed, as indicated by time stamps—ascertainable, for example, using a method that exists as part of the Java language—associated with a predefined beginning and end of execution of the program.

In one embodiment of the invention, a monitoring method monitors the position of the content on a display screen while the content is being displayed. The position is evaluated to produce monitoring information regarding the display of the content. Such evaluation can be accomplished, for example, by further obtaining information regarding the position of one or more other images on the display screen, and comparing the position of the content to the position of the one or more other images.

For example, in accordance with the above embodiment, the monitoring method can determine whether the content is not visible on the display screen, either because the content is occluded by the one or more other images, or because the content has been “scrolled” off of the display screen (hereafter, these two situations will be referred to together by saying that the content is “hidden”). Further, the monitoring method can determine whether the content is partially hidden, i.e., either partially occluded by the one or more other images, or partially scrolled off of the display screen. Moreover, the duration of time of each period during which the content is fully or partially hidden can be determined as such periods occur. The duration of time of unobstructed displays of the content can be determined as times when the view of the content is not either fully or partially hidden. Each of the durations can be reported directly and/or the total duration that the content is fully hidden, partially hidden, fully or partially hidden and/or unobstructed, respectively, can be reported.

Information regarding whether or not the displayed content is hidden can be useful for a variety of reasons. For example, such information indicates the amount of time that the displayed content was visible to the observer for observation. Additionally, this information can be used by the content provider to determine in which regions of a display screen his content is most likely to be unobstructed.

Whether the displayed content is hidden can be determined in any manner that is possible using the tools (e.g., supported programming language, operating system characteristics) associated with the computer network with which the invention is being used. One way of determining whether the displayed content is hidden using the above-described applet is to periodically declare that the content display (or a portion thereof) is invalid, i.e., the operating system is asked to redraw the content display, if necessary. If the operating system then makes a request to the applet to redraw the content display, then the content display is not hidden. However, if the operating system does not make a request to the applet to redraw the content display, then the content display is hidden.

The most complete information regarding whether the content display is hidden can be obtained by invalidating each discrete element of the content display (e.g., pixel) and determining whether the discrete element is hidden, in the manner described above. However, such an approach is computationally expensive and is generally not necessary to obtain useful and sufficiently accurate information regarding whether the content display is hidden. Preferably, then, only a portion of the content display, strategically selected, is evaluated in this manner.

For example, in one embodiment of the invention, each of the corners of the content display are invalidated and monitored for redrawing as described above. If all of the corners are redrawn, then the content display is determined to be unobstructed. If none of the corners are redrawn, then the content display is determined to be fully hidden. If at least one, but not all, of the corners are redrawn, then the display is determined to be partially hidden.

FIGS. 4A, 4B and 4C are simplified views of a display screen 400 including a content display 401 and other images 402, 403 and 404, illustrating an unobstructed, fully hidden, and partially hidden content display, respectively. In FIG. 4A, none of the corners 401a, 401b, 401c or 401d are covered by one of the other images 402, 403 and 404. Thus, after the corners 401a, 401b, 401c and 401d are invalidated, each is redrawn, and the content display 401 is (correctly, in this case) determined to be unobstructed. In FIG. 4B, each

of the corners **401a**, **401b**, **401c** and **401d** is covered by the image **402**. Thus, none of the corners **401a**, **401b**, **401c** and **401d** are redrawn after being invalidated, and the content display **401** is (again, correctly) determined to be fully hidden. In FIG. 4C, the corners **401c** and **401d** are covered by the image **402**, but the corners **401a** and **401b** are not. Thus, only the corners **401a** and **401b** are redrawn after invalidation of the corners **401a**, **401b**, **401c** and **401d**, and the content display **401** is (once again, correctly) determined to be partially hidden.

The above approach may not be accurate in all cases. FIG. 4D is a simplified view of the display screen **400** including the content display **400** and other images **402**, **403** and **404**, illustrating a partially hidden content display **401** that would be determined to be fully hidden according to the method of the invention detailed above. In FIG. 4D, the image **402** covers the corners **401c** and **401c**, the image **403** covers the corner **401a**, and the image **404** covers the corner **401b**. Thus, since none of the four corners **401a**, **401b**, **401c** and **401d** is redrawn after being invalidated, the content display **401** is determined to be fully hidden; however, as can be seen in FIG. 4D, this is not the case.

This problem can be alleviated by evaluating other discrete elements of the content display in addition to the corners. For example, discrete elements at the center of the upper and lower edges, and/or the right and left edges of the content display could be evaluated in addition to the corner pixels. FIG. 4E is a simplified view of the display screen **400** as shown in FIG. 4D, illustrating how another method of the invention can correctly determine the content display **401** to be partially hidden. In FIG. 4E, the upper edge center **401e** and the lower edge center **401f** of the content display **401** are also evaluated. The lower edge center **401f** is covered by the image **402**, while the upper edge center **401e** is not. Thus, after invalidation of the corners **401a**, **401b**, **401c** and **401d**, and the centers **401e** and **401f**, the upper edge center **401e** is redrawn, and the content display **401** is (correctly) determined to be partially hidden.

FIG. 4F is a simplified view of the display screen **400**, including the content display **401** and other images **402**, **403** and **404**, illustrating a partially hidden content display **401** that may be determined to be unobstructed according to a method of the invention. In FIG. 4F, none of the corners **401a**, **401b**, **401c** and **401d**, or the centers **401e** and **401f** are covered by the images **402**, **403** and **404**. Thus, after invalidation of the corners **401a**, **401b**, **401c** and **401d**, and the centers **401e** and **401f**, each is redrawn, and the content display **401** is determined to be unobstructed. However, as can be seen in FIG. 4F, this is not the case, since the image **403** is positioned in the middle of the content display **401**.

As illustrated in FIGS. 4D through 4F, while the evaluation of additional discrete elements of a content display does not eliminate the possibility of an inaccurate determination regarding whether the content display is hidden, it does reduce the likelihood of such occurrence. Generally, any number and configuration of discrete elements of a content display can be evaluated to reduce the possibility of an incorrect determination regarding whether the content display is hidden, so long as the associated computational cost does not become unacceptably high. Further, the above-described method for determining whether a content display is hidden is only one way in which such determination can be made.

As part of determining whether the content display is hidden, a time stamp is recorded each time there is a change in the "hidden state" of the content display. From these time stamps, the duration of each period of time that the content

display is unobstructed, partially hidden and fully hidden can be determined. From the duration of each period, total durations of time that the content display is unobstructed, partially hidden and fully hidden can also be determined.

It may also be possible, by appropriately configuring the discrete elements of the content display that are evaluated, to determine (though, typically, approximately), when the content display is partially hidden, the amount of the content display that is visible. It can be possible, too, when the content display is partially hidden, to give a qualitative description of the portion of the content display that is hidden (or visible), e.g., upper right corner, lower left corner.

When the monitoring method operates on a computer system having an event-driven operating environment, the monitoring method can monitor events as transmitted by the operating system to ascertain information regarding the content display. When the monitoring method is implemented as an applet that also displays the content, such monitoring can occur naturally, since only events concerning the content display are transmitted to the monitoring method. For example, the applet can use a pre-existing Java method (e.g., the method named `HandleEvent` in a current version of Java) to monitor events as transmitted by the operating system. Such event monitoring can be used to, for example, determine the number of times that an on-screen pointer (e.g., a mouse arrow or a cursor) entered an area defined by the content display. (The defined area can be related to the content display in any manner and can be, for example, the area in which the content is displayed, or an area somewhat smaller or larger than the area of the content display.) The operating system of the computer system displaying the content display typically monitors the position of the on-screen pointer and can identify in which region on the display screen the pointer is located. Thus, an applet configured to display content, as described above, can discern whether the pointer is located within the content display by monitoring an event that indicates that the pointer has entered the area defined by the content display. The monitoring method of the invention can use this information provided by the operating system to count the number of times that the on-screen pointer enters the area defined by the content display. The monitoring method can also determine when the on-screen pointer leaves the defined area after each entry, by monitoring another event that indicates that the pointer has exited the area defined by the content display. The time stamps associated with the entry into and exit from the defined area can be used to calculate the duration of time that the pointer was in the defined area for each entry into the defined area, as well as the total duration of time that the pointer was within the defined area. The monitoring method can also determine when the on-screen pointer is moving within the defined area, again by monitoring an event that indicates such pointer movement. The above-described information regarding the on-screen pointer position and movement relative to the content display may be useful in determining how attentive the observer was to the content, since an observer frequently watches the position of the on-screen pointer when viewing the display screen.

In another embodiment of the invention, the monitoring method monitors the change in time of a characteristic of the content display. The change in time of this characteristic is evaluated to produce monitoring information. The evaluation can be accomplished, for example, by further monitoring the change in time of a characteristic of the computer system used to display the content, and comparing the change in time of the characteristic of the content display to

the change in time of the characteristic of the computer system. Either of the two examples given immediately above (hiding of the content display and entry of a pointer into a defined area) are also examples of a monitoring method in accordance with this embodiment of the invention.

A monitoring method according to the invention can obtain a variety of other information, as well. For example, the monitoring method can obtain a time stamp (date and time of day) that indicates when the display of the content began. When the monitoring method is implemented by an applet written in Java, the time stamp can be obtained using a method that exists as part of the Java language.

Identifying information regarding the computer on which the content is displayed can also be obtained. The Internet Protocol (IP) address from which the request for the content was made, as well as an identification of the machine to which the content was transferred can be obtained. (There may not be a one-to-one correspondence between these two if, for example, the latter is a client computer of a system for which the former is a server computer.) Again, both the IP address and machine name can be obtained using a pre-existing Java method.

A monitoring method according to the invention can also determine if the user of the computer at the content display site selected (e.g., clicked with a mouse or pressed an appropriate keyboard key) a hyperlink within the area of the content display to end display of the current content display. Similar to the monitoring of the pointer location described above, an applet that implements a monitoring method of the invention can include a standard Java method (e.g., `HandleEvent`) that accepts events transmitted by the operating system. One of the events is the selection of a hyperlink. When such an event is reported, the monitoring method can so note.

As previously indicated, the above-described examples of monitoring information are merely illustrative of the types of monitoring information that can be obtained by a monitoring method according to the invention. Generally, a monitoring method according to the invention can make use of any method available in the computing environment, e.g., an operating system method, or a method that is part of a software framework, or that can be written in a computer programming language that can be used in the computing environment. For example, when the monitoring method is implemented by an applet written in Java, any existing Java method can be used to obtain information relevant to the display of the content to be monitored, either by using the method to change the state of the computer (e.g., the state of the display) on which the content is being displayed and monitoring the response of the computer (e.g., the method for monitoring whether content display is hidden, discussed above) or by retrieving information about the state of the computer (e.g., the method for monitoring entry of the pointer into the content display, discussed above). In particular, the monitoring of events as discussed above can be useful in discerning information about the content display.

A monitoring method according to the invention can also be used to ascertain information about an audio display. For example, if the content being monitored includes audio content that can only be displayed by selecting an appropriate user interface mechanism (e.g., a graphical pushbutton), a monitoring method according to the invention can determine whether that "event" is transmitted to the window represented by the content, indicating that the audio display was at least begun. Using a method as described

above for determining the duration of a content display, together with knowledge of the when the audio display was begun (using a time stamp as described above), the duration of the audio display can also be determined. It may also be possible to determine the volume at which the audio content is displayed, by appropriately monitoring the methods used to operate the audio display devices. These examples are merely illustrative. As can be appreciated, using any other method available in the computing environment, other information regarding an audio display can be determined.

A monitoring method according to the invention can also be used to explicitly (i.e., by presenting questions to observers that they can answer) acquire demographic information regarding the observers of the content being monitored. This could be implemented, for example, by including the instructions for asking such questions, the content of the questions and the instructions for storing the obtained demographic information in a computer program used to implement the monitoring method. Or, such instructions and question content could be stored in a separate file that is called and executed by the computer program that implements the monitoring method. Or, instructions for presenting the questions and storing the answers could be included as part of the computer program for implementing the monitoring method, and the content of the questions could be contained in a separate file that is accessed by the computer program. These latter two possibilities can be particularly advantageous, since they allow multiple sets of demographic questions to be presented to observers by the monitoring method, thus enabling the demographic questions to be tailored to the content being displayed or to the characteristics of the observer likely to view the content.

As described above, in accordance with the invention, monitoring information regarding the display of content can be obtained, then later reviewed and analyzed to enable conclusions to be drawn about how the content was displayed and, possibly, to enable deductions to be made about how the content was observed. In addition, monitoring information can be used to affect the display of a set of content. One way in which this can occur is for a set of content, or the manner in which the set of content is displayed, to be modified based upon review and analysis of monitoring information obtained from previous displays of the set of content (e.g., monitoring information regarding whether or not the content was hidden, or the frequency of display of the content at different times during the day or week, that may be used to determine the best location on a display screen or the best times, respectively, to display the content).

Another way in which monitoring information can be used to affect the display of a set of content is to use certain monitoring information obtained just before or during the display to cause the set of content to be displayed in a particular manner. For example, as discussed above, the IP address from which a request for a set of content emanated can be ascertained when the request is first received. It may be possible to associate characteristics of an observer or observers with an IP address from which a request for content has been received (because, for example, demographic information has previously been obtained as described above when a set of content was previously transferred to that IP address). Based upon the known characteristics associated with the IP address, an appropriate one of multiple versions of the requested set of content can be transferred for display, e.g., if it is known that the IP address corresponds to a content display site that is used by observers that speak a particular language, then text dis-

played aurally or visually can be displayed in that language. As another example, the duration of time that a set of content has been displayed can be determined, as discussed above, and the portion of the set of content that is being displayed changed as a function of that duration, e.g., the display of a set of content can begin with a particular video display and change to another video display after passage of a specified duration of time. As still another example, the portion of a set of content that is displayed can be varied based upon performance characteristics of the network over which the content is transferred. For example, the amount of time required to transfer data from the content provider site to the content display site can be monitored (by, for example, obtaining information from the log file regarding the size of transferred files and the amount of time required to transfer those files, as discussed above). The display of the content can then be controlled so that a moving video is displayed if the data transfer rate is above a predetermined magnitude and a still video is displayed if the data transfer rate is below the predetermined magnitude, the predetermined magnitude being chosen so that data rates below that magnitude are insufficiently fast to produce moving video of acceptable quality. The above examples are merely illustrative; other ways of using monitoring information to affect the display of a set of content are contemplated by the invention.

As described above, a monitoring method according to the invention can obtain monitoring information regarding the display of content. Of particular interest is the basic question of whether the content was displayed at all. As described above, a monitoring method according to the invention can make this determination. Some observers, however, have developed techniques for suppressing the display of particular content (e.g., advertisements). A monitoring method according to the invention can also increase the likelihood that particular content is displayed by conditioning certain other operation of the computer system that displays the content on the display of that particular content. For example, the content to be monitored can be presented as part of other content. Such presentation is common on, for example, the World Wide Web, where, for example, advertising content is frequently included as part of other content. A monitoring method according to the invention can condition the display of the other content on the display of the to-be-monitored content, e.g., the full content of a Web page cannot be viewed unless an advertisement included on the Web page is viewed. Moreover, the presentation of the other content can be conditioned on the display of the to-be-monitored content for a specified period of time. This can be particularly valuable when the to-be-monitored content does not appear automatically as part of the other content, but, rather, is only displayed in response to selection of an appropriate user interface mechanism (e.g., a graphical pushbutton) that is part of the other content.

In accordance with the above-described forced presentation of specified content, the detection of content suppression can be accomplished in any suitable manner. For example, it may be possible to detect the suppression technique being used. Or, the display of the content can be ordered so that the content that must be displayed ("required content") is displayed first; if the monitoring method detects that the content display site is displaying the other content without first having displayed the required content, then suppression of the required content has been detected. Upon detection of suppression of the required content, display of the other content is prevented and, if desired, a message indicating that fact can be displayed.

As indicated above, after monitoring information is obtained by a monitoring method according to the invention,

the monitoring information is transferred from the content display site **302** to a remote site. The remote site can be the content provider site **301** or another site that is part of the network. When a monitoring method according to the invention is implemented by a Java applet, the remote site is the content provider site **301**, since, currently, such applets can only communicate information to the network site from which they were transferred. However, in the future, such constraint may not exist; in that event, the remote site need not necessarily be the content provider site **301** even when a Java applet is used to implement a monitoring method according to the invention.

Generally, the monitoring information can be transferred to the remote site at any time. It may be desirable, for example, to cause the monitoring information to be transferred to the remote site immediately after the monitoring information is obtained, so that the monitoring information is accessible as quickly as possible. It may, alternatively, be desirable to store the monitoring information at the content display site, then transfer the monitoring information at a time when communication over the network communication line **303** is fastest and/or least expensive, e.g., at night.

The monitoring information can be communicated to a communication port that is different than the port from which the content and the monitoring instructions were transmitted to the content display site **302**. In that event, a special daemon that monitors such communication port for receipt of monitoring information is installed on the server computer at the content provider site. The daemon can be implemented as a conventional server daemon for monitoring data received by a server computer on a designated communication port. Communication of the monitoring information to a specially designated port can be useful to enable the monitoring data to be sent in any desired format in accordance with any desired protocol. For example, the monitoring data can be encrypted, as described below.

When the invention is implemented with the World Wide Web, it is also possible to transmit the monitoring data over the network using the communication channel monitored by the http daemon, i.e., by transmitting a request to the http daemon. Such transmission may be desirable for several reasons. For example, transmission of monitoring data to the http daemon eliminates the need to create, and supply to operators of the remote site to which the monitoring data is to be transferred (e.g., a Web page operator or, as described below, an application manager site operator), special software for receiving the monitoring data. Additionally, transmission of monitoring data by transmitting a request to the http daemon may be the only way to transfer the monitoring data to the remote site. This can be true, for example, when one or more client computers are served by a "proxy server" which mediates communication between the client computers and other sites on the network. The proxy server may not allow communication over a channel specially designated for transmitting monitoring data, but allow communication to the http daemon.

Transmission of monitoring data by making a request to the http daemon can be accomplished in a variety of ways. For example, an http request can be submitted for a file having a "name" that denotes the monitoring data in some way. Notwithstanding the spurious nature of the file request, the request is recorded in the http log file, from which the "name" can be retrieved to enable extraction of the monitoring data. Or, a request for execution of a CGI script can be transmitted, with the parameter of the CGI script request that specifies input to the script being specified to denote the monitoring data in some way. A computer program resident

on the computer system at the remote site can then implement a method that extracts the value of the input from the CGI script, and the monitoring data can be extracted from the value of the input. Other methods of using CGI scripts or http requests to transmit monitoring data to an http daemon are possible.

For security, it may be desirable to encrypt the monitoring data before it is transferred from the content display site **302** to a remote site. Any suitable encryption method can be used. For example, a public key encryption method, such as the well known RSA algorithm, can be used to encrypt the monitoring data. In general, the monitoring data (or other data transferred over a network in accordance with the invention) can be encrypted before any transmission of the data over a network (other examples of such data transmission are described below as part of the systems illustrated in FIGS. **5A**, **5B** and **5C**, and FIGS. **6A**, **6B**, **6C** and **6D**).

Once communicated to the remote site, the monitoring information can be stored in any appropriate database, as known to those skilled in the art of constructing and managing databases. The monitoring information can be presented for observation through a suitable user interface, such as a graphical user interface (GUI), in any desired format, e.g., graphs, bar charts, pie charts. The monitoring information stored in the database can also be subjected to further analysis if desired. For example, the total time that a content display is available to be viewed can be broken down into percentages of time that the content display was unobstructed, partially hidden and fully hidden. Or, the percentage of observers of a set of content that select a particular hyperlink while observing the content can be identified.

The monitoring information may be of interest not only to the content provider that provides the content for display, but to third parties as well. For example, if the content provided by the content provider includes an advertisement, the advertiser may be interested in the monitoring information regarding display of the content. The third party and the content provider may have conflicting interests in the substance of the monitoring information. For example, if the third party is paying the content provider to include the third party's content with the content provider's content, and the payment is based upon the amount of exposure of the third party's content to observers, the content provider has an interest in the monitoring information showing a large amount of exposure of the content, while the third party has an interest in the monitoring information showing a small amount of exposure. (Both parties, of course, can be simultaneously motivated by other interests, as well: for example, the third party may simply want the monitoring information to reflect accurately the amount of exposure of the content, so that they can use that information in assessing the effects of providing their content through the content provider.) If the monitoring information is transferred from the content display site to the content provider site, and unrestricted access to the monitoring information allowed at the content provider site, there may be no foolproof way to prevent the content provider from tampering with the substance of the monitoring information. This problem is particularly acute when a monitoring method according to the invention is embodied in a manner (e.g., by a Java applet), as discussed above, that necessitates that the monitoring information be transferred back to the content provider site.

FIGS. **5A**, **5B** and **5C** are simplified diagrams of a network illustrating operation of another embodiment of the invention. This embodiment of the invention provides a system configuration that can overcome the problem of

possible tampering with the substance of the monitoring information by the content provider. As in the system illustrated in FIGS. **3A**, **3B** and **3C**, a content display site **302** is linked over a network to a content provider site **301**. The network also includes an application manager site **501**. The content display site **302** and content provider site **301** can communicate with each other via the network communication line **303**, as described above, to enable transfer of content and monitoring instructions from the content provider site **301** to the content display site **302**. Alternatively, the content and monitoring instructions can be transferred to the content display site **302** from the application manager site **501** in response to a request received from the content provider site **301** upon receipt of the request from the content display site **302**. This latter implementation is illustrated in FIGS. **5A**, **5B** and **5C**. In such an implementation, the content provider site **301** needn't have either a computer program for implementing the monitoring method or a program for receiving monitoring data installed at the content provider site **301**, thus simplifying use of the invention for the content provider. Rather, the content provider need only have an open account (as discussed below) at the application manager site **501**.

In this embodiment of the invention, the monitoring information obtained at the content display site **302** is transferred to the application manager site **501**, either directly from the content display site **302** or indirectly via the content provider site **301**. If the latter, then the monitoring information can be received by the content provider site **301** and transferred to the application manager site **501** in a way that prevents access to the monitoring information at the content provider site **301**. For example, the monitoring information could be encrypted at the content display site **302** before transfer to the content provider site **301**, the decryption method being available only at the application manager site **501**. Or, the monitoring information could be immediately transferred to the application manager site **501** after being received at the content provider site **301**. Once received at the application manager site **501**, access to the monitoring information can be administered by the (neutral) application manager so that the monitoring information can not be modified by any of the parties having an interest in the information, thus ensuring the integrity of the monitoring information.

In a typical implementation, multiple sets of content will be provided from multiple content provider sites, and each set of content will be displayed by multiple content display sites. A set of monitoring information will be recorded for each display of each of the multiple sets of content and transferred to the application manager site for storage in a database that is implemented on a computer at the application manager site. Each set of monitoring information must be identified as corresponding to the set of content for which the monitoring information was obtained, so that monitoring information can be appropriately stored in a database to enable later retrieval of the monitoring information for that set of content. When a monitoring method according to the invention is implemented for use with the World Wide Web, this can be accomplished by appropriate specification of a parameter included in a computer program written in html used to implement a monitoring method, as discussed above. Example 3 below illustrates how Example 1 discussed above can be modified to make such specification (the "Account" parameter).

```

<applet code="//AppMgr.com/AdInsert.class" width=230 height=33>
<param name="image" value="images/southwest.gif">
5 <param name="href" value="http://www.swa.com/">
<param name="Account" value="9004560093">
</applet>

```

The database residing on the computer at the application manager site can also be used, for example, to store account information about the content provider site from which the content display is provided.

In the embodiment of the invention illustrated in FIGS. 5A, 5B and 5C, a user interface (e.g., GUI) can be provided on the content provider site computer to enable the owner (or representative) of the content provider site to access monitoring information stored at the application manager site regarding content displays provided by the content provider site. Such an interface can also be configured to enable the content provider to create a new account on the application manager computer, authorize payments for use of the monitoring system of the invention, and request particular analysis or presentation of obtained monitoring information. Other functions can also be provided in such an interface, as desirable.

It is also possible that there be multiple application manager sites. Typically, monitoring information for each content display will be designated for storage on a particular one of the application manager sites. Such designation can be included as a parameter specification in a computer program used to implement the monitoring information as discussed above.

As discussed above, the content provided by a content provider can be tailored according to any specified criteria. Further, the content provider may periodically update the content. Additionally, third parties may want to provide their content with that of a content provider. These third parties may also have multiple sets of specially tailored content that are updated periodically. The management of such multiple sets of content by a content provider at the content provider site can become undesirably complex and may overtax the available bandwidth for transmission of data to and from the content provider site.

FIGS. 6A, 6B, 6C and 6D are simplified diagrams of a network illustrating operation of still another embodiment of the invention. This embodiment of the invention provides a system configuration that can enable updated and/or tailored secondary content provided by a secondary content provider to be transferred to a primary content display site for use with primary content supplied by a primary content provider without the problems identified above with existing such systems, as discussed in more detail below. This embodiment of the invention also can enable all of the functionality described above for the system illustrated in FIGS. 5A, 5B and 5C. In the embodiment of the invention shown in FIGS. 6A, 6B, 6C and 6D, a content display site 302, a primary content provider site 602 and an application manager site 501 are linked to each other over a network and can communicate with each other as described above. The network also includes a secondary content provider site 601. As shown in FIG. 6A, in this embodiment of the invention, secondary content can be provided from the secondary content provider site 601 to the application manager site 501 and stored thereat. As shown in FIG. 6B, whenever secondary content is provided to the application manager site 501, the application manager causes the content to both be stored

at the application manager site 501 and transferred to all content provider sites, e.g., content provider site 602, that provide that secondary content with their primary content. When a request for the primary content that includes such secondary content is received by the primary content provider site 602 from the content display site 302 (FIG. 6C), the primary content provider site 602 is able to immediately (i.e., without necessity to retrieve the content from another network site or request that the content be provided to the content display site from another network site) provide both the primary and secondary content to the content display site 302, as shown in FIG. 6D.

As can be appreciated, the management of both primary and secondary content can become quite burdensome when many sets, and/or many versions of sets, of secondary content and/or primary content are being provided from the primary content provider site. Management of continual updates to these sets of content data exacerbates this burden. By storing secondary content (and, if desired, primary content) at the application manager site 501, the system of FIGS. 6A, 6B, 6C and 6D relieves the primary content provider of the burden of managing such content. However, because the application manager causes the content to be stored at the content provider site 602, the content can be provided to the content display site 302 from the content provider site 602, rather than the application manager site 501, thus leaving control of bandwidth management with the primary content provider so that the primary content provider can ensure that a system that adequately addresses the bandwidth requirements of the content provided from the primary content provider site 602 is in place. This is an important consideration for the primary content provider since requesters of content from the primary content provider will hold the primary content provider responsible for the performance characteristics (e.g., speed) associated with the provision of that content. The system of FIGS. 6A, 6B, 6C and 6D, then, relieves the primary content provider of the need to manage the storage of content, while reserving control over the provision of that content to the content provider.

In this embodiment of the invention a user interface (e.g., GUI) can be provided at both the primary content provider site 602 and the application manager site 501. The primary content provider user interface can provide the same functionality as described above with respect to FIGS. 5A, 5B and 5C. Additionally, the primary content provider user interface can enable the content provider to select available secondary content for possible inclusion with that content provider's primary content. Such selection can also include specification of terms upon which the primary content provider wishes to include the secondary content. Selection of secondary content does not automatically cause the secondary content to be included with the primary content provider's content, but, rather, causes a request for such inclusion to be made (e.g., via the secondary content provider user interface, described below) to the secondary content provider. Upon acceptance by the secondary content provider, the secondary content can be included with the primary content. The secondary content provider user interface can enable the secondary content provider to select a primary content provider site with which to include the secondary content provider's content. Again, such selection can be made together with specification of the terms of such inclusion; the selection causes a request for inclusion to be made (e.g., via the primary content provider user interface) to the primary content provider. The secondary content user interface can also provide functionality similar to that

described above with respect to FIGS. 5A, 5B and 5C. As will be readily appreciated by those skilled in the art, other functions can also be provided in the primary content provider and secondary content provider user interfaces, as desirable. The embodiment of the invention shown in FIGS. 6A, 6B, 6C and 6D facilitates interaction between the primary content provider site 602 and the secondary content provider site 601 to enable a secondary content provider to easily and flexibly provide content to a primary content provider in a manner that enables both the primary and secondary content providers to exercise control over the provision of content.

As described above, monitoring instructions and content can be embodied by an applet that executes at the content display site. In the system of FIGS. 5A, 5B and 5C or the system of FIGS. 6A, 6B, 6C and 6D, the use of such an applet can advantageously dovetail with an implementation of those systems in which the applet is transferred to the content display site from the application manager site. This is because an applet must return the monitoring information to the network site from which the applet was transferred: thus, the monitoring information is transferred directly to the neutral application manager site. Too, when the monitoring instructions and content are transferred from the application manager site to the content display site, the use of monitoring information to tailor the content provided to the content display site, as discussed above, can also be easily implemented.

Various embodiments of the invention have been described. The descriptions are intended to be illustrative, not limitative. Thus, it will be apparent to one skilled in the art that certain modifications may be made to the invention as described above without departing from the scope of the claims set out below. For example, though the invention has been described above as it particularly applies to monitoring the display of content disseminated over the World Wide Web, the invention can generally be used to monitor the display of content disseminated over any computer network. Additionally, though an implementation of the invention has been described in which aspects of the Java programming language are used, it is to be understood that invention is not limited to such implementation; other programming languages could be used having other features and characteristics (e.g., the language need not be an object-oriented language as is Java).

I claim:

1. A system for monitoring display of content by a computer system, comprising:

means for monitoring the position of a content display on a display screen of the computer system; and

means for evaluating the position of the content display on the display screen to produce monitoring information regarding display of the content.

2. A system as in claim 1, further comprising means for monitoring the position of one or more other images on the display screen, and wherein the means for evaluating further comprises means for comparing the position of the content display to the position of the one or more other images to produce the monitoring information.

3. A system as in claim 2, wherein the means for comparing further comprises means for determining whether the content display is hidden by the one or more other images.

4. A system as in claim 3, wherein the means for comparing further comprises:

means for determining the duration of each time that the content display is hidden by the one or more images; and

and

means for determining the duration of each time that the content display is not hidden by the one or more images.

5. A system as in claim 3, wherein the means for comparing further comprises:

means for determining the total duration of time that the content display is hidden by the one or more images; and

means for determining the total duration of time that the content display is not hidden by the one or more images.

6. A system as in claim 3, wherein the means for determining further comprises means for determining whether the content display is fully hidden or partially hidden by the one or more other images.

7. A system as in claim 6, wherein the means for comparing further comprises:

means for determining the duration of each time that the content display is fully hidden by the one or more images;

means for determining the duration of each time that the content display is partially hidden by the one or more images; and

means for determining the duration of each time that the content display is not hidden by the one or more images.

8. A system as in claim 6, wherein the means for comparing further comprises:

means for determining the total duration of time that the content display is fully hidden by the one or more images;

means for determining the total duration of time that the content display is partially hidden by the one or more images; and

means for determining the total duration of time that the content display is not hidden by the one or more images.

9. A system as in claim 2, wherein:

one of the other images is a pointer; and

the means for comparing further comprises means for determining the number of times that the pointer entered an area defined by the content display.

10. A system as in claim 1, wherein at least some of the content being monitored comprises a graphical display.

11. A system for monitoring display of content by a computer system, comprising:

means for monitoring the change in time of a characteristic of a content display; and

means for evaluating the change in time of the characteristic of the content display to produce monitoring information regarding display of the content.

12. A system as in claim 11, further comprising means for monitoring the change in time of a characteristic of the computer system, and wherein the means for evaluating further comprises means for comparing the change in time of the characteristic of the content display to the change in time of the characteristic of the computer system to produce the monitoring information.

13. A system for monitoring display of content by a computer system, wherein the content is displayed in response to an instruction that is provided from a source external to the computer system and the system for monitoring, comprising:

means for ascertaining the beginning of a display of the content;

27

means for ascertaining the end of a display of the content;
and

means for monitoring display of the content, wherein:
the means for monitoring begins operating when the
beginning of a display of the content is ascertained;
and
the means for monitoring stops operating when the end
of a display of the content is ascertained.

14. A system as in claim 13, wherein the instruction is
provided by a user of the computer system.

15. A system as in claim 13, wherein the means for
monitoring the display of the content further comprises:

means for monitoring the position of the content display
on a display screen of the computer system; and

means for evaluating the position of the content display
on the display screen to produce monitoring informa-
tion regarding display of the content.

16. A system as in claim 13, wherein the means for
monitoring the display of the content further comprises:

means for monitoring the change in time of a character-
istic of the content display; and

means for evaluating the change in time of the charac-
teristic of the content display to produce monitoring
information regarding display of the content.

17. A system as in claim 13, wherein the means for
monitoring further comprises means for determining the
duration of the display of the content.

18. A system for monitoring display at a content display
site of content that is provided by a content provider site
over a network to the content display site, comprising:

means for monitoring display of the content to produce
monitoring information regarding display of the con-
tent; and

means for transferring the means for monitoring from the
content provider site to the content display site in
response to the transfer of content from a content
provider site.

19. A system as in claim 18, wherein the means for
monitoring is transferred to the content provider site
together with the content.

20. A system as in claim 18, further comprising means for
transferring the monitoring information to a remote site that
is part of the network.

21. A system as in claim 20, wherein the remote site is the
content provider site from which the content was transferred
to the content display site.

22. A system as in claim 21, further comprising means for
transferring the monitoring information from the content
provider site to a second remote site.

23. A system as in claim 22, wherein the second remote
site is adapted to receive and store monitoring information
from a plurality of content provider sites.

24. A system as in claim 21, wherein monitoring infor-
mation is transferred from the content display site to the
content provider site via a communication means that is
different from the communication means used to transfer the
content from the content provider site to the content display
site.

25. A system as in claim 18, wherein new content can be
transferred to the content display site in response to selection
of a portion of the content currently being displayed at the
content display site.

26. A system as in claim 18, wherein the monitoring
information includes information from which conclusions
may be deduced regarding the observation of the content by
an observer.

28

27. A system as in claim 18, wherein the means for
monitoring further comprises:

means for monitoring the position of the content display
on a display screen at the content provider site; and
means for evaluating the position of the content display
on the display screen to produce monitoring informa-
tion.

28. A system as in claim 18, wherein the means for
monitoring further comprises:

means for monitoring the change in time of a character-
istic of the content display; and

means for evaluating the change in time of the charac-
teristic of the content display to produce monitoring
information.

29. A system as in claim 18, wherein:

the content is displayed in response to an instruction that
is provided from a source external to the network and
the system for monitoring; and

the means for monitoring further comprises:

means for ascertaining the beginning of a display of the
content; and

means for ascertaining the end of a display of the
content, wherein:

the means for monitoring begins operating when the
beginning of a display of the content is ascer-
tained; and

the means for monitoring stops operating when the
end of a display of the content is ascertained.

30. A system for monitoring display at a content display
site of content that is provided by a content provider site
over a network to the content display site, wherein the
network operates according to a protocol that enables new
content to be transferred to a content display site in response
to selection of a portion of content currently being displayed
at the content display site, the system comprising:

means for monitoring the display of content to produce
monitoring information regarding display of the con-
tent; and

means for transferring the means for monitoring from the
content provider site to the content display site so that
the means for monitoring operates at the content dis-
play site.

31. A system as in claim 30, wherein at least some of the
content being monitored comprises a graphical display.

32. A system as in claim 30, wherein the means for
monitoring further comprises:

means for monitoring the position of the content display
on a display screen at the content provider site; and

means for evaluating the position of the content display
on the display screen to produce the monitoring infor-
mation.

33. A system as in claim 30, wherein the means for
monitoring further comprises:

means for monitoring the change in time of a character-
istic of the content display; and

means for evaluating the change in time of the charac-
teristic of the content display to produce monitoring
information regarding display of the content.

34. A system as in claim 30, wherein:

the content is displayed in response to an instruction that
is provided from a source external to the network and
the system for monitoring; and

the means for monitoring further comprises:

means for ascertaining the beginning of a display of the
content; and

29

means for ascertaining the end of a display of the content, wherein:

the means for monitoring begins operating when the beginning of a display of the content is ascertained; and

the means for monitoring stops operating when the end of a display of the content is ascertained.

35. A system as in claim 30, further comprising means for transferring the monitoring information to a remote site that is part of the network.

36. A system for monitoring display at a content display site of content that is provided by a content provider site over a network to the content display site, comprising:

means for monitoring display of the content to produce monitoring information regarding display of the content; and

means for transferring the monitoring information from the content display site to a remote site of the network that is different from the content provider site.

37. A system as in claim 36, wherein the means for transferring the monitoring information further comprises:

means for transferring the monitoring information from the content display site to the content provider site; and

means for transferring the monitoring information from the content provider site to the remote site, wherein the monitoring information cannot be stored at the content provider site, or accessed or manipulated before transfer to the remote site.

38. A system as in claim 36, further comprising means for storing monitoring information at the remote site.

39. A system as in claim 38, wherein the means for storing monitoring information at the remote site is adapted to enable storage of monitoring information regarding the display of a plurality of sets of content.

40. A system as in claim 38, wherein:

the network includes a plurality of content provider sites; and

the means for storing monitoring information at the remote site is adapted to enable storage of monitoring information regarding the display of content provided from the plurality of content provider sites.

41. A system as in claim 38, further comprising means for accessing the monitoring information stored at the remote site from a site on the network other than the remote site, such that a user at the other site can interact with the monitoring information, but cannot modify the monitoring information.

42. A system for providing updated and/or tailored content from a first content provider site to a content display site, comprising:

means for transferring content from the first content provider site to a second content provider site;

means for recognizing the presence of updated and/or tailored content stored at the first content provider site, wherein the means for transferring content transfers the updated and/or tailored content from the first content provider site to the second content provider site in response to recognition of the presence of updated and/or tailored content at the first content provider site; and

means for transferring content from the second content provider site to the content display site when such content is requested by the content display site.

43. A system as in claim 42, further comprising an application manager system and wherein the means for transferring content from the first content provider site to the second content provider site further comprises:

30

means for transferring content from the first content provider site to the application manager system; and means for transferring content from the application manager system to the second content provider site.

44. A system as in claim 42, wherein the means for transferring content transfers the updated content from the first content provider site to the second content provider site automatically upon recognition of the updated and/or tailored content at the first content provider site.

45. A system as in claim 42, wherein:

there are a plurality of second content provider sites; and the means for transferring content transfers the updated and/or tailored content from the first content provider site to each of the plurality of second content provider sites in response to recognition of the updated and/or tailored content at the first content provider site.

46. A system as in claim 42, wherein there are a plurality of content provider sites to which content can be transferred from the first content provider site, and further comprising means for enabling a user at the first content provider site to select one or more of the plurality of content provider sites to be second content provider sites.

47. A system as in claim 42, wherein there are a plurality of content provider sites from which content can be transferred to the second content provider site, and further comprising means for enabling a user at the second content provider site to select one or more of the plurality of content provider sites to be first content provider sites, wherein only a content provider site so selected can transfer content to the second content provider site.

48. A computer readable medium encoded with one or more computer programs for enabling monitoring of the display of content by a computer system, comprising:

instructions for monitoring the position of a content display on a display screen of the computer system; and instructions for evaluating the position of the content display on the display screen to produce monitoring information regarding display of the content.

49. A computer readable medium as in claim 48, further comprising instructions for monitoring the position of one or more other images on the display screen, and wherein the instructions for evaluating further comprise instructions for comparing the position of the content display to the position of the one or more other images to produce the monitoring information.

50. A computer readable medium as in claim 49, wherein the instructions for comparing further comprise instructions for determining whether the content display is hidden by the one or more other images.

51. A computer readable medium as in claim 50, wherein the instructions for comparing further comprise:

instructions for determining the duration of each time that the content display is hidden by the one or more images; and

instructions for determining the duration of each time that the content display is not hidden by the one or more images.

52. A computer readable medium as in claim 50, wherein the instructions for comparing further comprise:

instructions for determining the total duration of time that the content display is hidden by the one or more images; and

instructions for determining the total duration of time that the content display is not hidden by the one or more images.

53. A computer readable medium as in claim 50, wherein the instructions for determining further comprise instruc-

31

tions for determining whether the content display is fully hidden or partially hidden by the one or more other images.

54. A computer readable medium as in claim 53, wherein the instructions for comparing further comprise:

instructions for determining the duration of each time that the content display is fully hidden by the one or more images;

instructions for determining the duration of each time that the content display is partially hidden by the one or more images; and

instructions for determining the duration of each time that the content display is not hidden by the one or more images.

55. A computer readable medium as in claim 53, wherein the instructions for comparing further comprise:

instructions for determining the total duration of time that the content display is fully hidden by the one or more images;

instructions for determining the total duration of time that the content display is partially hidden by the one or more images; and

instructions for determining the total duration of time that the content display is not hidden by the one or more images.

56. A computer readable medium as in claim 49, wherein: one of the other images is a pointer; and

the instructions for comparing further comprise instructions for determining the number of times that the pointer entered an area defined by the content display.

57. A computer readable medium encoded with one or more computer programs for enabling monitoring of the display of content by a computer system, comprising:

instructions for monitoring the change in time of a characteristic of a content display; and

instructions for evaluating the change in time of the characteristic of the content display to produce monitoring information regarding display of the content.

58. A computer readable medium as in claim 57, further comprising instructions for monitoring the change in time of a characteristic of the computer system, and wherein the instructions for evaluating further comprise instructions for comparing the change in time of the characteristic of the content display to the change in time of the characteristic of the computer system to produce the monitoring information.

59. A computer readable medium encoded with one or more computer programs for enabling monitoring of display of content by a computer system, for use with a computer system in which content is displayed in response to a content display instruction that is provided from a source external to the computer system and not part of the monitoring computer program or programs, comprising:

instructions for ascertaining the beginning of a display of content;

32

instructions for ascertaining the end of a display of the content; and

instructions for monitoring display of the content, wherein:

the instructions for monitoring begin executing when the beginning of a display of the content is ascertained; and

the instructions for monitoring stop executing when the end of a display of the content is ascertained.

60. A computer readable medium as in claim 59, wherein the content display instruction is provided by a user of the computer system.

61. A computer readable medium as in claim 59, wherein the instructions for monitoring the display of the content further comprise:

instructions for monitoring the position of the content display on a display screen of the computer system; and

instructions for evaluating the position of the content display on the display screen to produce monitoring information regarding display of the content.

62. A computer readable medium as in claim 59, wherein the instructions for monitoring the display of the content further comprise:

instructions for monitoring the change in time of a characteristic of the content display; and

instructions for evaluating the change in time of the characteristic of the content display to produce monitoring information regarding display of the content.

63. A computer readable medium as in claim 59, wherein the instructions for monitoring further comprise instructions for determining the duration of the display of the content.

64. A computer readable medium encoded with one or more computer programs for enabling monitoring of display of content by a computer system, comprising:

instructions for causing content to be displayed by the computer system; and

instructions for monitoring display of content by the computer system to produce monitoring information regarding the display of the content, wherein the monitoring instructions are integrated with the display instructions such that execution of the display instructions causes execution of the monitoring instructions.

65. A computer readable medium encoded with one or more computer programs for enabling monitoring of display of content at a content display site, comprising:

instructions, adapted for use at the content display site, for monitoring display of content at the content display site to produce monitoring information regarding display of the content; and

instructions for receiving monitoring information from the content display site.

* * * * *

APPENDIX B-12



US007089304B2

(12) **United States Patent**
Graham

(10) **Patent No.:** **US 7,089,304 B2**

(45) **Date of Patent:** **Aug. 8, 2006**

(54) **METERED INTERNET USAGE**

(75) Inventor: **John C. Graham**, Palo Alto, CA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 739 days.

(21) Appl. No.: **09/943,766**

(22) Filed: **Aug. 30, 2001**

(65) **Prior Publication Data**

US 2003/0046409 A1 Mar. 6, 2003

(51) **Int. Cl.**

G06F 15/173 (2006.01)

G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/224**; 709/203; 709/223; 709/229

(58) **Field of Classification Search** 709/229, 709/225, 224

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | | |
|-----------|------|---------|--------------------|-------|---------|
| 5,987,424 | A * | 11/1999 | Nakamura | | 705/14 |
| 6,018,619 | A * | 1/2000 | Allard et al. | | 709/224 |
| 6,055,575 | A * | 4/2000 | Paulsen et al. | | 709/229 |
| 6,092,191 | A * | 7/2000 | Shimbo et al. | | 713/153 |
| 6,119,227 | A * | 9/2000 | Mao | | 713/171 |
| 6,170,075 | B1 * | 1/2001 | Schuster et al. | | 714/776 |
| 6,275,471 | B1 * | 8/2001 | Bushmitch et al. | | 370/248 |
| 6,289,451 | B1 * | 9/2001 | Dice | | 713/168 |
| 6,615,258 | B1 * | 9/2003 | Barry et al. | | 709/223 |
| 6,651,099 | B1 * | 11/2003 | Dietz et al. | | 709/224 |
| 6,778,509 | B1 * | 8/2004 | Ravishankar et al. | | 370/322 |

OTHER PUBLICATIONS

Nakamura, M.; Sato, M.; Hamada, T.; A Pricing and Accounting Software Architecture for QOS Guaranteed Ser-

vices on a Multidomain Network, *Electronics and Communications in Japan, Part 1 (Communications)*, vol. 84, No. 3, pp. 38-47.

Rizzo, M.; Briscoe, B.; Tassel, J.; Damianakis, K.; A Dynamic Pricing Framework to Support a Scalable, Usage-Based Charging Model for Packet-Switched Networks, *Active Networks, First International Working Conference, IWAN '99 Proceedings*, pp. 48-59.

Anand, S.S.; Kasturi, K.; Sriram, G.; Accounting Architecture for Cellular Networks, *1996 IEEE International Conference on Personal Wireless Communications Proceedings and Exhibition—Future Access*(Cat. No. 96TH8165), pp. 184-189.

(Continued)

Primary Examiner—Zarni Maung

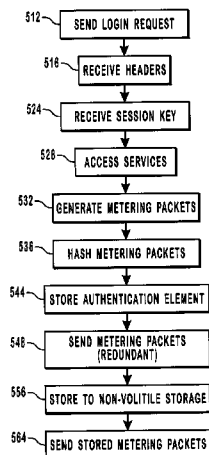
Assistant Examiner—Kamal Divecha

(74) *Attorney, Agent, or Firm*—Workman Nydegger

(57) **ABSTRACT**

Methods, systems and computer program products for tracking a client's usage of one or more services provided by one or more servers. A client generates and sends one or more metering packets to a census service. Each metering packet includes a time element indicating the client's usage of the one or more services. The time element may include a charged time portion and a free time portion. An authentication element may be included with each metering packet so that the census service can determine whether or not a given metering packet is genuine. A login service communicates to the client whether or not usage should be tracked and indicates a time interval to expire between subsequent metering packets. A session identifier in each metering packet allows multiple sessions to be tracked simultaneously. Upon receiving metering packets, the census service discards redundant metering packets and updates a usage database accordingly.

40 Claims, 6 Drawing Sheets



OTHER PUBLICATIONS

Kumar, B.; Effect of Packet Losses on End-User Cost in Internetworks With Usage Based Charging, *Computer Communication Review*, vol. 23, No. 2, pp. 9-15.

Estrin, D.; Zhang, L.; Design Considerations for Usage Accounting and Feedback in Internetworks, *Integrated Network Management, II. Proceedings of the IFIP TC6/WG6.6 Second International Symposium*, pp. 719-733.

* cited by examiner

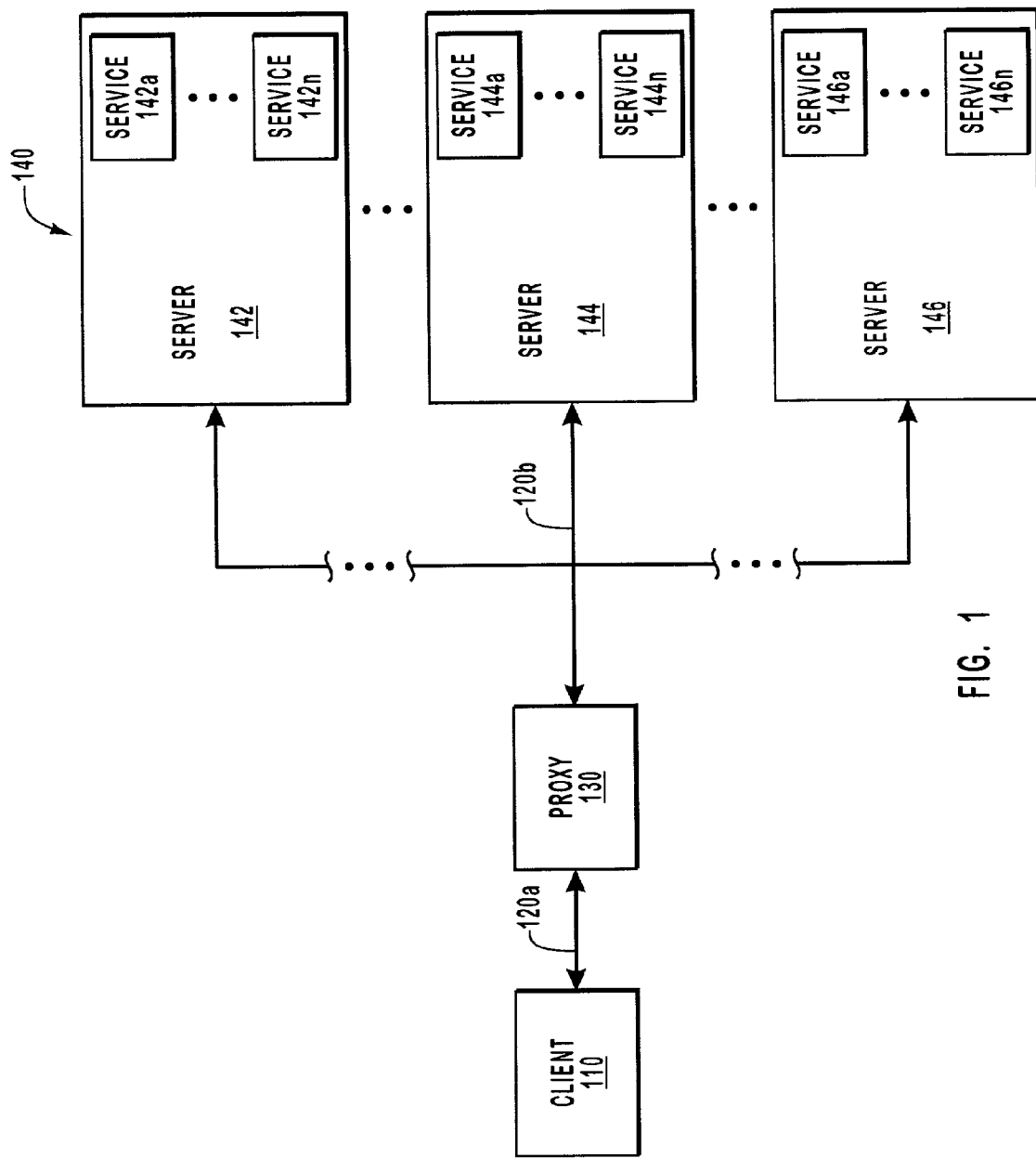


FIG. 1

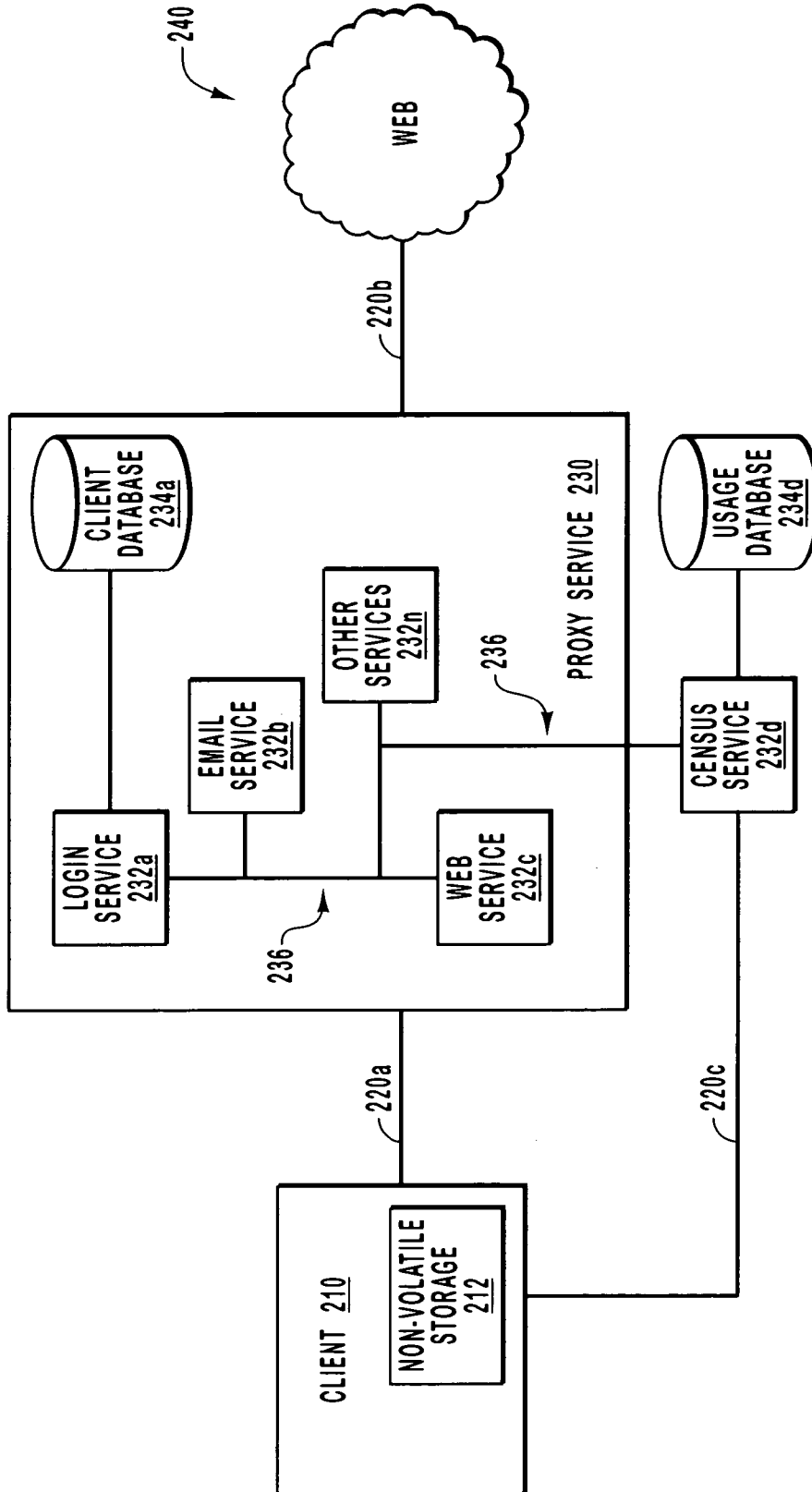


FIG. 2

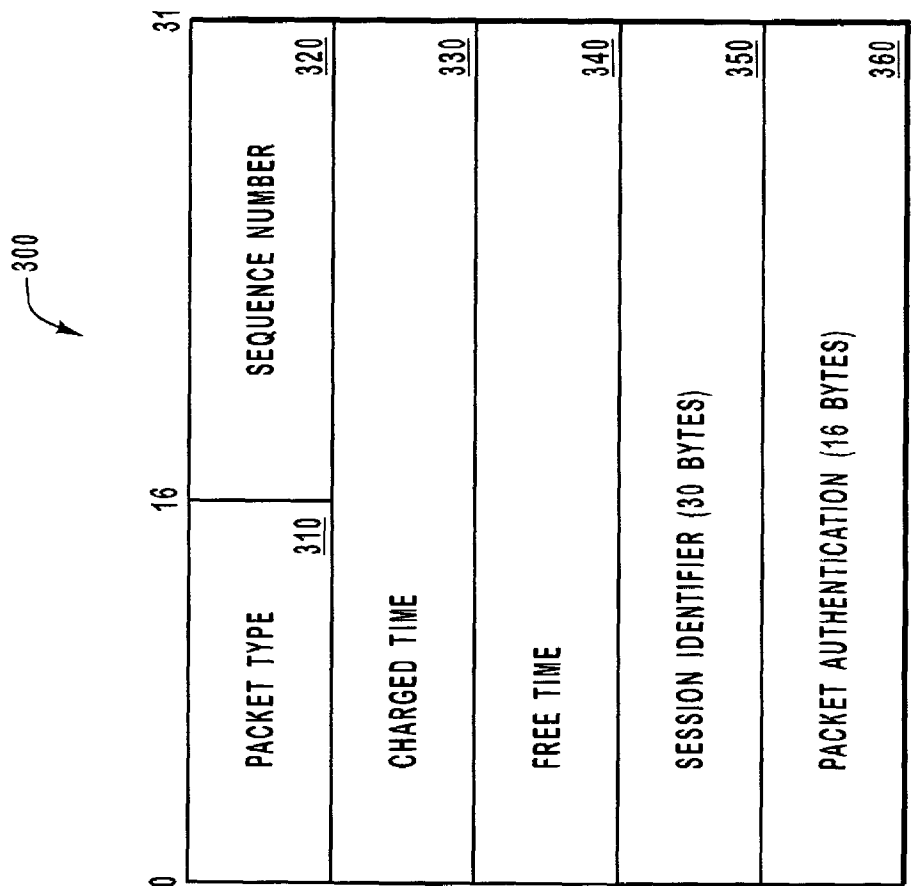


FIG. 3

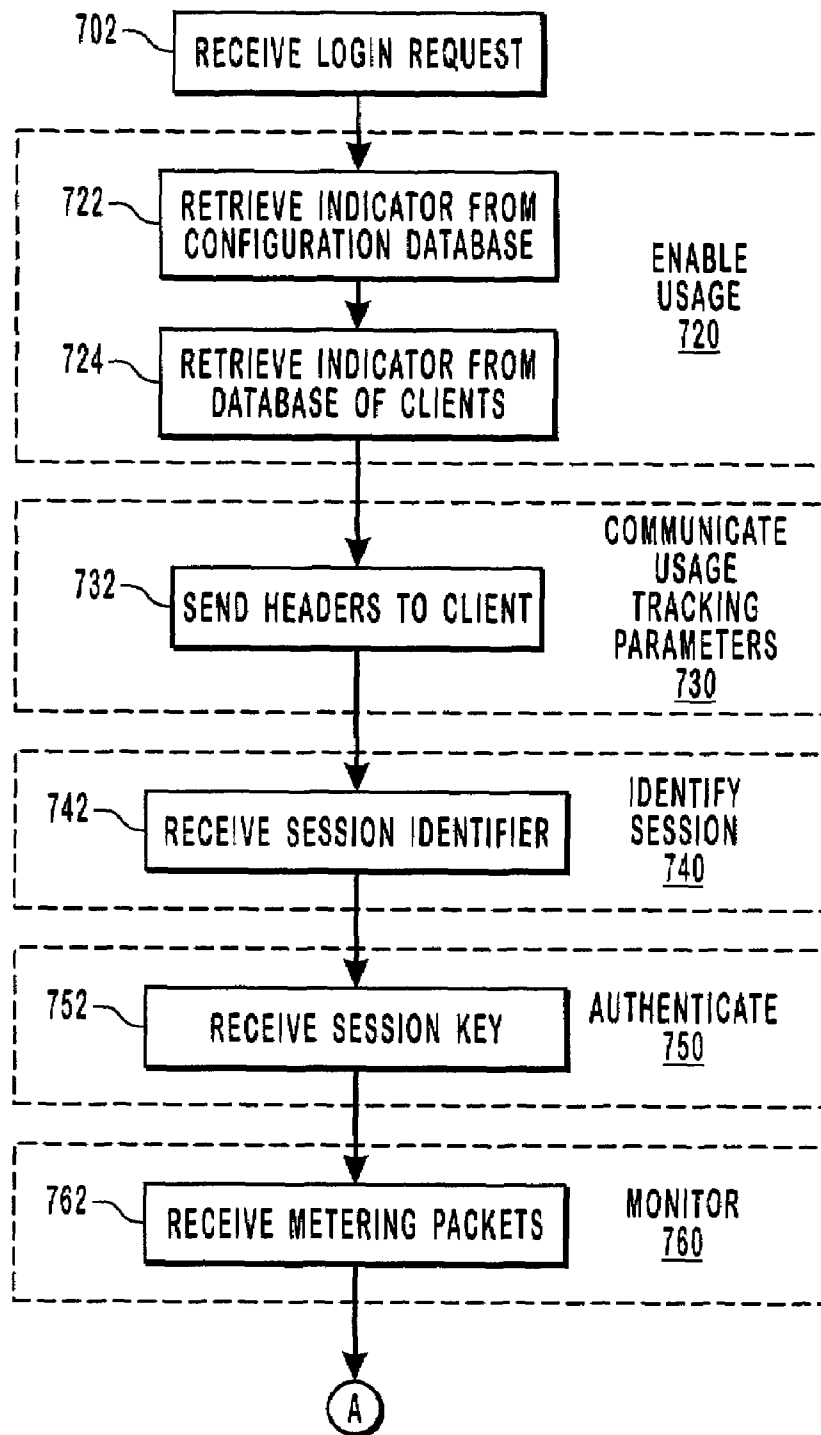


FIG 4A

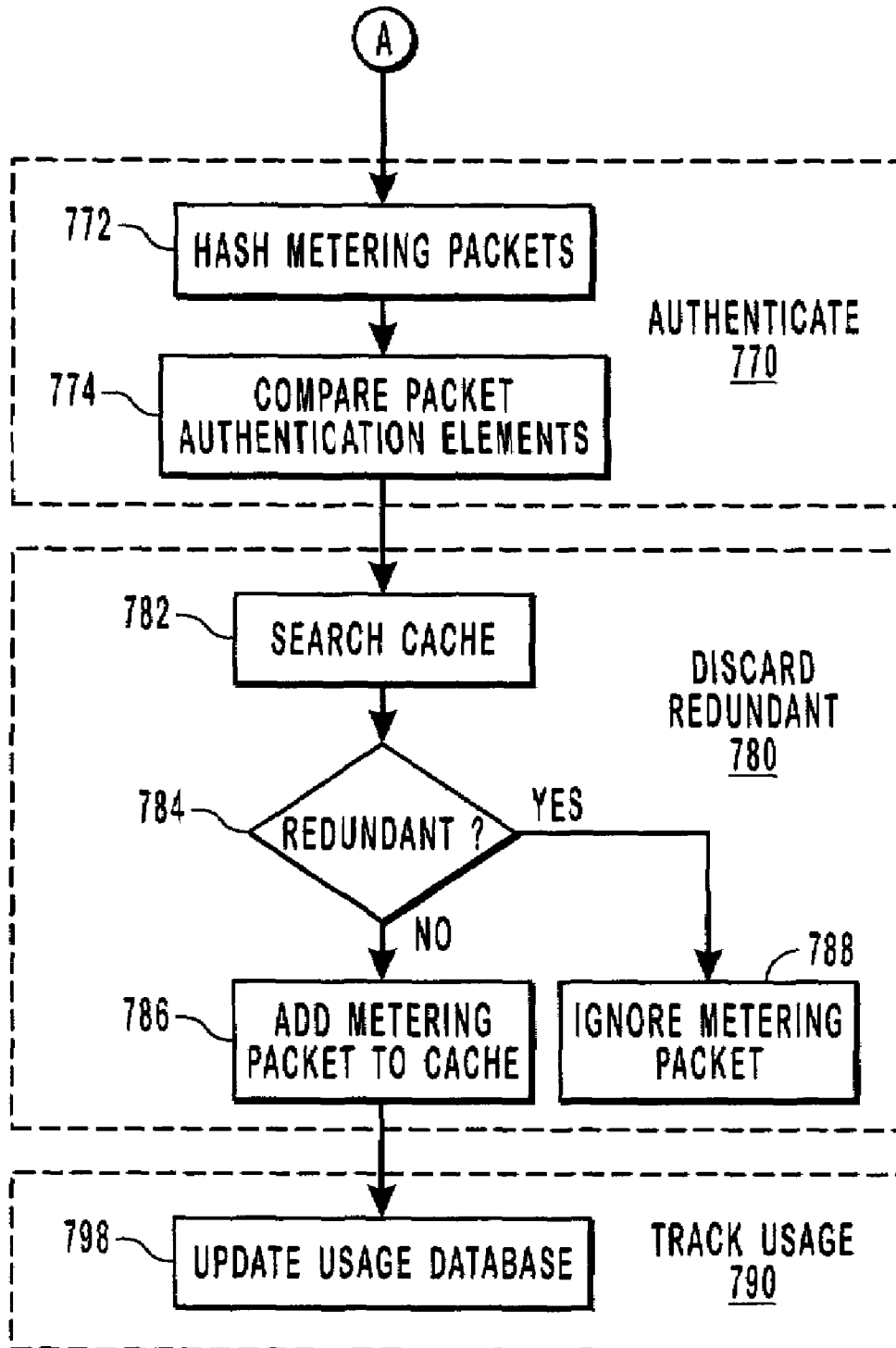


FIG 4B

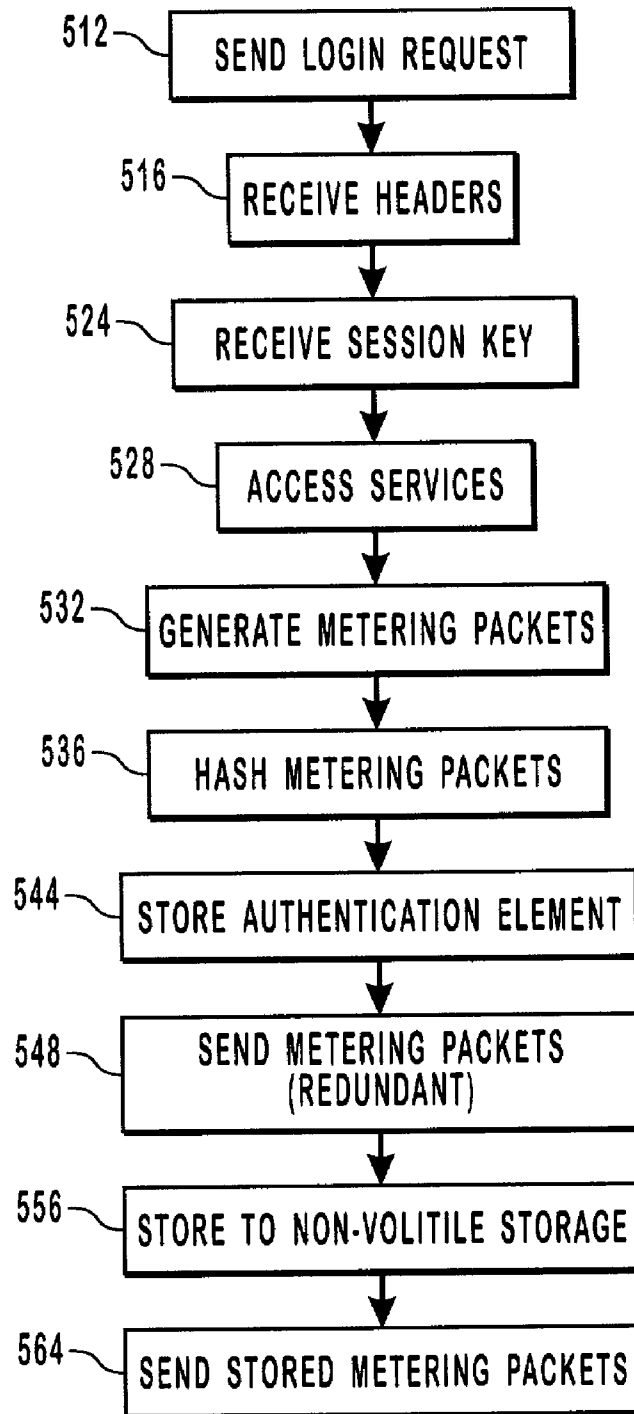


FIG 5

1

METERED INTERNET USAGECROSS-REFERENCE TO RELATED
APPLICATIONS

N/A

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates to the usage of computer services. More specifically, the present invention relates to methods, systems and computer program products for tracking a client's usage of one or more services provided by one or more servers.

2. Background and Related Art

With the exception of certain educational institutions and governmental entities, most access to the Internet is indirect. That is, rather than being directly connected to the Internet, most access the Internet through some intermediary, known as an Internet Service Provider or ISP. An ISP may provide various levels of service depending on the particular needs of its customers. For example, individual customers may access the Internet through a dialup telephone line, a broadband cable, or perhaps a broadband wireless connection. Many individual customers typically share an ISP's connection resources, at least to some extent. In contrast, business users often prefer a dedicated ISP connection with a fairly constant bandwidth.

An ISP may provide various "points of presence" for connecting to the Internet. Depending on the ISP, these points of presence may include local telephone numbers, toll or toll free telephone access numbers, cable systems, microwave stations, etc. Cable systems and microwave systems are local by nature, but it is generally a significant advantage for an ISP to offer local telephone access to keep costs as low as possible, both for the ISP and its customers.

Although the number of computers with Internet access has grown tremendously, competition among ISPs can be quite fierce. As a result, ISPs often employ various service plans with aggressive pricing strategies to attract consumers, including businesses and individuals alike. Most service plans fall into one of two broad categories: (i) unlimited access for a fixed fee, and (ii) a certain number of hours for a fixed fee, with additional hours being billed as used. In either case, an ISP server is traditionally responsible for tracking a client's usage, if necessary.

However, traditional tracking suffers from at least two significant problems. First, tracking each client connected to an ISP may impose a considerable processing burden on a server. This burden may be especially pronounced where an ISP has offered only unlimited access for a fixed fee, but would like to begin providing a reduced service level that requires usage tracking. In such a case, tracking may require upgrading to more powerful servers in order to avoid an overall performance reduction. Given the rather competitive nature of the ISP market, much of the benefit gained in offering a variety of service levels may be substantially offset by the corresponding increased costs and/or diminished capacity.

The second problem is at least somewhat related to the first. In some circumstances, it may be desirable to distinguish between various types of access. More particularly, an ISP may wish to charge for access to one type of service, whereas access to another type of service may be without charge. Tracking this level of detail at the ISP, as compared to simply tracking raw connection time, imposes yet further

2

performance loads on the ISP's computing resources. Here again, the tradeoffs, in terms of benefits versus associated costs, may be undesirable or even prohibitive.

In contrast to an ISP's computer resources, a client's computing resources may be comparatively underutilized. Furthermore, the overhead associated with having an individual client track its own usage of services is likely to represent a much less significant performance problem for the client. Whereas server-based tracking concerns the usage of each and every connected client, client-based tracking concerns the usage of an individual client, or perhaps a cluster of clients. As such, client-based tracking allows a substantial portion of the processing load to be borne by the client. While some type of centralized server tracking component may be useful in receiving and correlating usage information from individual clients, the server computer resources for implementing client-based tracking are likely to be significantly less than would be required in a comparable, substantially server-based, tracking implementation. Therefore, methods, systems and computer program products for tracking a client's usage of server services are desired.

BRIEF SUMMARY OF THE INVENTION

The present invention uses one or more client-generated metering packets to track a client's usage of one or more services provided by one or more servers. Metering packets may be generated by a client and sent to and received by a server over regular periodic intervals. Each metering packet includes a time element that indicates the client's usage of the provided services. The time element may include a charged time portion for access to services that incurs an access charge and a free time portion for access to services that does not incur an access charge. Metering packets also may include other elements, such as a packet type element, a sequence number element, a session identifier element, a packet authentication element, etc. Among other things, a packet type element may be used to indicate whether a packet is for an active session currently in progress or for a session that is ending.

When a communication protocol that does not guarantee delivery is used, sending redundant metering packets increases the likelihood that the information contained within any particular metering packet is received, even if one or more packets are lost. However, if packet delivery is successful, sending more than one metering packet with the same information may be redundant. A sequence number element may help identify any redundant metering packets that are received. Then, redundant metering packets may be discarded rather than processed to conserve computing resources. For example, a usage database may be updated to reflect the tracking information contained within a metering packet. Prior to updating the usage database, a cache of previously received metering packets may be searched, and if a metering packet with the same sequence number is found in the cache, a newly received metering packet can be identified as redundant and ignored. Otherwise, the usage database is updated with the tracking information in the newly received metering packet and the newly received metering packet is added to the cache. The sequence number element also may help determine if some metering packets have not been received.

Where multiple sessions are tracked, a session identifier element may be used to link a particular metering packet with a particular session. An authentication element may be used to assure that any given metering packet is genuine. For

3

example, a session key may be associated with a specific session. Some of the tracking information within a metering packet and the session key may be hashed to generate an authentication element that is included within the metering packet. When the metering packet is received, the same tracking information and session key are hashed at the receiving end. Comparing the authentication element generated at the receiving end with the authentication element included within the metering packet determines whether or not the metering packet is genuine.

Upon receiving a login request from a client, a login service may check configuration data to determine if the client should track usage. The configuration data may include an indicator from a configuration database that indicates whether or not usage should be tracked for all clients who login and an indicator from a database of clients that indicates whether or not usage should be tracked for a particular client. Configuration data may be extended to indicate whether or not usage should be tracked for a particular session. If the configuration data dictates that usage should be tracked, the login service communicates to the client that the client should track its usage of the one or more services provided by one or more servers. For example, the login service may send one or more headers to the client to indicate that the client should track usage and communicate various usage tracking parameters.

While tracking usage, a client may terminate access to the services provided by the servers in any of a variety of ways, including hanging up, timing out, powering off, changing users, and logging off. In many circumstances, the client is able to send session-ending metering packets to indicate that a particular session is terminated. For example, when timing out, changing users or logging off, the client usually continues to operate and can send the appropriate session-ending packets without incident. However, in other circumstances, such as hanging up or powering off, the client may discontinue operation and be unable to send one or more session-end packets. Furthermore, metering packets may be sent over an unreliable transport protocol that does not guarantee delivery. Regardless of the motivation, the client may store metering information in non-volatile memory and then send the stored metering information in a subsequent session. This helps assure that usage tracking remains accurate even when there is some uncertainty as to whether or not a particular metering packet is received.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered as limiting its scope, the invention

4

will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates an exemplary system that provides a suitable operating environment for the present invention;

FIG. 2 illustrates an exemplary system according to the present invention;

FIG. 3 is a block diagram showing the data structure of an exemplary metering packet according to the present invention;

FIGS. 4A and 4B are flow diagrams, from the perspective of a server, describing various acts and steps for methods according to the present invention; and

FIG. 5 is a flow diagram, from the perspective of a client, describing various acts of methods according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention extends to methods, systems, and computer program products for tracking a client's usage of one or more services provided by one or more servers. By employing one or more client-generated metering packets to track the client's usage of the one or more services, the present invention avoids the otherwise substantial processing burden imposed by substantially server-based approaches. The client-generated metering packets also provide increased flexibility and enhanced accuracy, in terms of what usage incurs an access charge, how various types of session terminations are handled, and in determining when a session actually ends. The embodiments of the present invention may comprise a special-purpose or general-purpose computer including various computer hardware, as discussed in greater detail below.

Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Computer-executable instructions comprise, for example, instructions and data which cause a general-purpose computer, special-purpose computer, or special-purpose processing device to perform a certain function or group of functions. Such computer-readable media can be any available media that can be accessed by a general-purpose or special-purpose computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, CD-ROM or other optical disc storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general-purpose or special-purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media.

FIG. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by computers in network environments. Generally, program modules include rou-

tines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represents examples of corresponding acts for implementing the functions described in such steps.

Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention also may be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention may include client 110, proxy 130, and servers 140. Each of client 110, proxy 130, and servers 140 may be implemented as a general- or special-purpose computing device. Such a computing device may include a processing unit, system memory, and a system bus that couples various system components to the processing unit. A processing unit in combination with program code means is one example of a processor means. System memory may include read only memory (ROM), random access memory (RAM), non-volatile RAM, and/or any other type of memory.

The computing devices also may include a magnetic hard disk drive, a disk drive for reading from or writing to a removable media, such as magnetic disks, optical discs, or other magnetic/optical media. The drives are connected to the system bus by one or more drive interfaces. Drives and/or interfaces for other types of computer readable media for storing data also may be present, including magnetic cassettes, flash memory cards, digital versatile disks, Bernoulli cartridges, RAMs, ROMs, and the like. The drives, their associated computer-readable media, and certain types of memory may provide non-volatile storage of computer-executable instructions, data structures, program modules and other data for the computing devices. Program code means comprising one or more program modules may be stored at each of the computer devices, including an operating system, one or more application programs, one or more services, other program modules, and program data.

Client 110, proxy 130, and servers 140 operate in a networked environment using network connections 120a and 120b to communicate with each other. The network connections 120a and 120b depicted in FIG. 1 may comprise a local area network and/or a wide area network (WAN). Such networking environments are commonplace in office-wide or enterprise-wide computer networks, intranets and the Internet. In a LAN networking environment, network connections 120a and 120 may include a network interface or adapter, a modem, a wireless link, or other means for establishing communications over a network, such as the Internet. Network connections 120a and 120b are examples of communication means. It will be appreciated that the network connections shown are exemplary and other means for communicating over a network may be used.

In general, client 110 accesses one or more of the services 142a–142n provided by server 142, services 144a–144n provided by server 144, and services 146a–146n provided by server 146 through network connections 120a and 120b and proxy 130. It should be noted that the terms client, proxy, and server are not mutually exclusive and should be interpreted broadly. A client consumes resources, a server provides resources, and a proxy operates on behalf of either a client or a server. In many circumstances the designations of client, proxy, and server apply for a particular time and then change. For example, a client at one time may be a proxy or a server at another, and so on. It should also be noted that each of the servers 140 may be a cluster of servers, and that each individual server may provide one or more services or a particular service may be implemented across one or more servers. The present invention does not require any particular configuration.

Turning next to FIG. 2, tracking client's 210 usage of one or more services provided by proxy service 230 will be described in greater detail. Communication between client 210 and proxy service 230, within proxy service 230 over network connections 236, and between proxy service 230 and either web 240 or census service 232d, follows a request/response protocol. The HyperText Transfer Protocol ("HTTP") is one type of well-known request/response protocol. Communication between client 210 and census service 232d may use some other communication protocol, such as User Datagram Protocol ("UDP"). However, the present invention is in no way limited to the use of any particular communication protocol or any particular network topology.

Proxy service 230 includes a login service 232a. In one embodiment, login service 232a is a Unix daemon. Logging in to proxy service 230 is accomplished in two stages. During the first stage, a secure session is established between client 210 and proxy service 230 so that sensitive information may be exchanged. After the first login stage, the login service 232a checks a configuration database (not shown) to determine if usage tracking is enabled for clients attempting to login. The login service 232a also checks a client database 234a to determine if usage tracking is enabled for a particular client, such as client 210. For certain types of access, usage tracking may not be enabled.

If usage tracking is enabled, the login service 232a creates a unique session identifier for the new session with client 210. Those of skill in the art will recognize that the present invention may be particularly useful in environments where sessions are relatively short. For example, mobile devices, such as personal digital assistants and cellular telephones, which tend to have short sessions (due to airtime expenses and/or other factors), may be more likely to benefit from the present invention, than other devices, such as personal computers where sessions may be several hours or days. Nevertheless, the present invention is not necessarily limited to any particular session duration or environment.

A session key is negotiated between the login service and the client to enable secure communication between client and login service. As described in greater detail below, the login service sends a hash of the session key to a census service for use in authenticating metering packets. Hashing the session key provides an extra measure of security because the session key is not communicated to other systems—only the client and the login service know the key's value. As used in this application, the term "session key" should be interpreted broadly to encompass any value suitable for authentication. In one embodiment, the session key is hashed using a Message Digest 5 ("MD5") hash. The

login service **232a** also sends one or more headers to the client. The one or more headers include usage tracking parameters, such as an indication that the client should track usage, the unique session identifier, a metering interval indicating how frequently the client should send metering packets, etc. The configuration database may be used to configure the metering interval. In one embodiment, the metering interval is expressed as a number of seconds.

There are two types of metering packets: a session-in-progress metering packet and a session-ending metering packet. The session-in-progress indicates that a session continues to be active, whereas a session-ending packet indicates that a session has terminated. It should be noted that the present invention is not necessarily limited to any particular type of metering packets. When the time interval expires or when a session ends, client **210** sends one or more metering packets over network connection **220a**. Redundant metering packets may be sent to increase the probability that they are received if an unreliable communication protocol is used. In one embodiment, three metering packets are sent using UDP. Each metering packet includes a sequence number so that redundant metering packets may be discarded.

Login service **232a** sends the unique session identifier and the MD5 hash of the session key to census service **232d** over network connections **236**. Census service **232d** receives the metering packets from client **210** over network connection **220c** and uses the session identifier to track the client's usage of the services provided by proxy service **230**, such as email service **232b**, web service **232c**, and other services **232n**. As metering packets are received, the census service checks a cache of received metering packets so that redundant packets are not reflected in usage database **234d**. Usage database **234d** is one example of usage means for tracking at least one client's usage of one or more services. The size of the cache is configurable. In one embodiment, the census service is implemented as a Unix daemon.

The census service **232d** is relatively simple. It does not maintain any state information for any client connections. Furthermore, policy decisions regarding valid sessions and billing are made in post-processing the usage database **234d**. However, if a session-in-progress metering packet is received for an unknown session, an error is generated. Redundant metering packets are discarded to avoid unnecessary updates to usage database **234d**. The census service **232d** employs a master-children architecture. A master process accepts all requests from the login service and processes them upon receipt. The master process also accepts the metering packets from client **210** and dispatches them to the children for processing. The number of children is configurable and therefore enhances scalability.

Note that proxy service **230** also provides access to the web **240** (e.g., the World Wide Web) over network connection **220b**. This access may be through web service **232c** or may be directly between client **210** and web **240**. For example, in one embodiment, secure connections between client **210** and web **240** bypass proxy service **230**. Note that in these circumstances, it would not be practical for server-based usage tracking to monitor client's **210** secure access to web **240**.

Hanging up, timing out, powering off, changing users, and logging off trigger client **210** to send one or more end-of-session metering packets. Because the session terminates, client **210** also stores the end-of-session metering packet in non-volatile storage **212**. When power is restored, client **210** reconnects, or another user logs in, client **210** restores the prior session data (including the end-of-session metering

packet) from non-volatile storage **212** and sends the prior session data to the login service **232a**. Upon receiving the prior session data, login service **232a** sends the session identifier for the new session and the prior session data to reinforce the end-of-session metering packets that were sent previously, but were not guaranteed to arrive. As noted earlier, in one embodiment metering packets are sent using UDP and even if sent, may not be delivered.

Census service **232d** also may authenticate metering packets to determine whether or not each packet is genuine. Authentication may be accomplished by hashing at least a portion of each packet and the session key, and sending the hash value with the packet. For example, the client first hashes the session key so that the client and the census service **232d** each have the same hash of the session key. (Recall that census service **232d** received a hash of session key along with the session identifier from login service **232a**.) The client then hashes the metering information in a metering packet and the hash of the session key to produce the hash value that can be used by the census service **232d** to authenticate the metering packet. Upon receipt, the census service **232d** performs a similar hash and compares the results with the hash value sent with the packet. If the two hash values do not match, the packet is not genuine. In one embodiment, the well-known MD5 algorithm, with a basic key known only to the client **210** and login service **232a**, is used to generate the hash value. However, the present invention is not limited to any particular hashing algorithm or authentication scheme.

It should be noted that a hacker looking at metering packets would not be able to deduce much. First, even if all metering information is transmitted as cleartext, the hacker will not be able to associate a particular session identifier with a specific client. Furthermore, without the session key, the hacker will not be able to generate a correct hash value for altered or created metering packets. As a result, the metering packets and census service are not susceptible to a man-in-the-middle-attack.

FIG. 3 is a block diagram showing the data structure of an exemplary metering packet **300** according to the present invention. The metering packet is 58 bytes long and includes a packet type element **310**, a sequence number element **320**, a time element that includes a charged time portion **330** and a free time portion **340**, a session identifier element **350**, and a packet authentication element **360**. Note that the packet type element **310** and sequence number element **320** are two bytes each, the change time portion **330** and free time portion **340** are four bytes each, the session identifier element **350** is 30 bytes, and the packet authentication element **360** is 16 bytes. Of course, the present invention is not necessarily limited to any particular metering packet size, content, or layout.

It should be noted that the charge time portion **330** and the free time portion **340** offer significant flexibility in billing client **210**. It may be desirable for proxy service **230** to provide some access to one or more services without charge. Tracking this level of detail in a substantially server-based implementation may impose a significant processing burden on a server and thereby erode much of the benefit provided by offering free time. Those of skill in the art will recognize that a free time portion may not be needed if only the amount of time to charge is of interest. For example, in some embodiments it may be desirable to track how much free time one or more client use, whereas in another embodiment, only the amount of time to charge is relevant.

It also should be noted here, that the client tracks its own usage. In particular, the client determines what access falls

within charge time portion 330 and what access falls within free time portion 340. In one embodiment, any access initiated automatically by the client, without user intervention, is accounted for in free time portion 340. For example, the client may initiate an automated download to receive a software update or other information. Nevertheless, the present invention is not necessarily limited to the use of any particular criteria in determining which access should be accounted for in the charge time portion 330 and which access should be accounted for in the free time portion 340.

The present invention also may be described in terms of methods comprising functional steps and/or non-functional acts. The following is a description of acts and steps that may be performed in practicing the present invention. Usually, functional steps describe the invention in terms of results that are accomplished, whereas non-functional acts describe more specific actions for achieving a particular result. Although the functional steps and non-functional acts may be described or claimed in a particular order, the present invention is not necessarily limited to any particular ordering or combination of the acts and/or steps.

FIGS. 4A and 4B are flow diagrams, from the perspective of a server, describing various acts and steps for methods according to the present invention. The present invention may include an act of receiving (702) a login request from a client. A step for enabling (720) usage tracking may include an act of retrieving (722) an indicator from a configuration database indicating that usage should be tracked for all clients attempting to login and an act of retrieving (724) an indicator from a database of clients indicating that usage should be tracked for a particular client. An act of sending (732) one or more headers to a client may achieve the result of communicating (730) one or more usage tracking parameters to the client, including at least one of (i) an indication that the client should track usage, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the client should send metering packets.

An act of receiving a session identifier (742) may achieve the result of identifying (740) one or more sessions through which a client accesses one or more services provided by one or more servers. A step for authenticating (750 and 770) may include acts of receiving (752) a session key associated with one or more sessions; an act of hashing (772) at least a portion of each metering packet and the corresponding session key to generate an authentication element; and, an act of comparing (774) the generated authentication element with a packet authentication element included with each metering packet to determine whether or not each packet is genuine.

As noted above, in one embodiment a census service receives the session identifier and a hash of the session key from a login service at about the same time that one or more headers are sent to a client. Although not shown, the present invention also may include an act of receiving metering packets that correspond to a previously terminated session. As described with respect to FIG. 5, a client may store metering information from one session in non-volatile memory and send the stored metering information in a subsequent session. In one embodiment, the client sends the stored metering information to the login service and the login service forwards the stored metering information to the census service. Among other things, this allows the census service to accept the metering information without independently authenticating it. As a result, it is not necessary for the census service to maintain session identifiers and session keys indefinitely.

A step for monitoring (760) one or more metering packets may be accomplished by an act of receiving (762) one or

more metering packets from a client, wherein each of the one or more metering packets includes a time element indicating the client's usage of the one or more services. A step for discarding (780) one or more redundant metering packets may include an act of, prior to updating a usage database, searching (782) a cache of at least one received metering packet; an act of, if a copy of a particular metering packet is found in the cache, identifying ("yes" branch of 784) the particular metering packet as redundant and not updating the usage database based on the particular metering packet or in other words ignoring (788) the particular metering packet; and, an act of, if a copy of the particular metering packet is not found in the cache ("no" branch of 784), adding (786) the particular metering packet to the cache. A step for tracking (790) a client's usage of one or more services may be achieved by an act of updating (798) a usage database based on one or more metering packets.

FIG. 5 is a flow diagram, from the perspective of a client, describing various acts of methods according to the present invention. The present invention may include an act of sending (512) a login request to a login service; an act of receiving (516) one or more headers from the login service including at least one of (i) an indication that a client should track usage of one or more services provided by one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the client should send metering packets; an act of receiving (524) a session key associated with the one or more sessions (whether received in a header or in some other way); an act of accessing (528), through the one or more sessions created in response to the login request, at least one of the one or more services provided by the one or more servers; and, generating (532) one or more metering packets, wherein each of the one or more metering packets includes a time element indicating the client's usage of the one or more services.

The present invention also may include an act of hashing (536) at least a portion of each metering packet to generate an authentication element; an act of storing (544) each authentication element in the corresponding metering packet; an act of sending (548) the one or more metering packets (possibly redundant) to a census service; an act of storing (556) metering information in non-volatile memory; and, an act of sending (564) the stored metering information to the census service in a subsequent session.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. In a computer network that comprises one or more servers providing one or more services to at least one client, and wherein the at least one client accesses the one or more services through the one or more servers during a plurality of sessions created in response to a login request from the at least one client, with at least some of the plurality of sessions occurring simultaneously, and wherein access to the one or more services during a particular session includes at least one of a charged time portion and a free time portion, a method of tracking the at least one client usage of the one or more services during each session the method comprising acts of:

receiving at one of the one or more servers one or more metering packets from the at least one client, each of the one or more metering packets being generated at the at least one client and each metering packet being used

11

at the at least one client to store data for tracking usage of one or more services during each session, and each metering packet comprising a data structure for storing the following data:

- a session identifier element that links a particular metering packet with a particular session;
- a time element indicating the at least one client usage of the one or more services, the time element comprising a charged time portion and a free time portion, wherein the charged time portion corresponds to access to one or more services that incurs an access charge, and wherein the free time portion corresponding to access to one or more services that does not incur an access charge; and
- a sequence number element; and the one or more servers updating a usage database based on the received one or more metering packets by using the sequence number element to determine whether each received metering packet is redundant of any prior metering packet already stored in the usage database, and if so, discarding it, and if not, then

storing each received metering packet that is not redundant in the usage database in order to store the data contained in each received metering packet that is not redundant, and from which it can be determined from the time element whether the at least client usage of the one or more services during the particular session for that received metering packet is a charged time portion or a free time portion.

2. A method as recited in claim 1, wherein a plurality of metering packets are received over regular, periodic intervals.

3. A method as recited in claim 1, wherein each of the one or more metering packets is one of a session-ending metering packet and a session-in-progress metering packet.

4. A method as recited in claim 1, further comprising acts of:

- receiving a session key associated with the one or more sessions;
- hashing at least a portion of each metering packet and the corresponding session key to generate an authentication element; and
- comparing the generated authentication element with a packet authentication element included with each metering packet to determine whether or not each packet is genuine.

5. A method as recited in claim 4, wherein a login service receives the login request from and negotiates a given session key with the at least one client, and wherein a census service receives the one or more metering packets, the method further comprising an act of the login service sending a hash of the given session key and a session identifier to the census service, such that the received session key is the hash of the given session key.

6. A method as recited in claim 5, further comprising: retrieving an indicator from a configuration database indicating that usage should be tracked for all clients attempting to login.

7. A method as recited in claim 1, wherein a plurality of metering packets are received and wherein one or more of the plurality of received metering packets are redundant, the method further comprising acts of:

- prior to updating the usage database, searching a cache of at least one received metering packet;
- if a copy of a particular metering packet is found in the cache, identifying the particular metering packet as

12

redundant and not updating the usage database based on the particular metering packet; and
if a copy of the particular metering packet is not found in the cache, adding the particular metering packet to the cache and updating the usage database based on the particular metering packet.

8. A method as recited in claim 7, wherein each metering packet comprises a session identifier element and a sequence number element, and wherein finding the particular metering packet in the cache is based on comparing the session identifier element and the sequence number element that are included with each metering packet.

9. A method as recited in claim 1, wherein each metering packet further comprises (i) a packet type element, (ii) a sequence number element, (iii) a session identifier element, and (iv) a packet authentication element.

10. A method as recited in claim 1, further comprising an act of sending one or more headers to the at least one client, wherein the one or more headers include at least one of (i) an indication that the at least one client should track usage of the one or more services provided by the one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the at least one client should send metering packets.

11. In a computer network that comprises one or more servers providing one or more services to at least one client, and wherein the at least one client accesses the one or more services through the one or more servers during a plurality of sessions created in response to a login request from the at least one client, with at least some of the plurality of sessions occurring simultaneously, and wherein access to the one or more services during a particular session includes at least one of a charged time portion and a free time portion, a method of tracking the at least one client usage of the one or more services during each session, the method comprising acts of:

in response to a login request received at one server of the one or more servers from the at least one client, a step for communicating from said one server to the at least one client usage tracking parameters;

thereafter a step for one or more metering packets being generated at the at least one client,

each metering packet being used at the at least one client to store data for tracking usage of one or more services during each session, and each metering packet comprising a data structure for storing the following data:

- a session identifier element that links a particular metering packet with a particular session; and
- a time element indicating the at least one client usage of the one or more services, the time element comprising a charged time portion and a free time portion, wherein the charged time portion corresponds to access to one or more services that incurs an access charge, and wherein the free time portion corresponding to access to one or more services that does not incur an access charge;

said one server performing a step for identifying one or more sessions through which the at least one client has accessed the one or more services;

the one server performing a step for monitoring metering packets that are received from the at least one client; and

the one server performing a step for tracking the at least one client usage of the one or more services during each session based on the received one or more metering packets in order to store data from which it can be determined whether the at least one client usage of the

one server performing a step for monitoring metering packets that are received from the at least one client; and

the one server performing a step for tracking the at least one client usage of the one or more services during each session based on the received one or more metering packets in order to store data from which it can be determined whether the at least one client usage of the

13

one or more services during each session is a charged time portion or a free time portion.

12. A method as recited in claim 11, wherein a plurality of metering packets includes both a session-ending metering packet and a session-in-progress metering packet.

13. A method as recited in claim 11, further comprising a step for authenticating the one or more metering packets.

14. A method as recited in claim 11, further comprising a step for enabling usage tracking in at least one of a configuration database and a database of clients.

15. A method as recited in claim 11, wherein a plurality of received metering packets are redundant, the method further comprising a step for discarding the one or more of the plurality of received metering packets that are redundant.

16. A method as recited in claim 11, wherein each metering packet further comprises (i) a packet type element, (ii) a sequence number element, (iii) a session identifier element, and (iv) a packet authentication element.

17. A method as recited in claim 11, further comprising a step for communicating one or more usage tracking parameters to the at least one client, wherein the one or more usage tracking parameters include at least one of (i) an indication that the at least one client should track usage of the one or more services provided by the one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the at least one client should send metering packets.

18. A computer program product for implementing, in a computer network that comprises one or more servers providing one or more services to at least one client, and wherein the at least one client accesses the one or more services through the one or more servers during a plurality of sessions created in response to a login request from the at least one client, with at least some of the plurality of sessions occurring simultaneously, and wherein access to the one or more services during a particular session includes at least one of a charged time portion and a free time portion, a method of tracking the at least one client usage of the one or more services during each session, the computer program product comprising a computer readable medium for carrying machine-executable instructions that implement the method, and the method comprising:

in response to a login request received at one server of the one or more servers from the at least one client, a step for communicating from said one server to the at least one client usage tracking parameters;

thereafter a step for one or more metering packets being generated at the at least one client,

each metering packet being used at the at least one client to store data for tracking usage of one or more services during each session, and each metering packet comprising a data structure for storing the following data: a session identifier element that links a particular metering packet with a particular session; and

a time element indicating the at least one client usage of the one or more services, the time element comprising a charged time portion and a free time portion, wherein the charged time portion corresponds to access to one or more services that incurs an access charge, and wherein the free time portion corresponding to access to one or more services that does not incur an access charge;

said one server performing a step for identifying one or more sessions through which the at least one client has accessed the one or more services;

14

the one server performing a step for monitoring metering packets that are received from the at least one client; and

the one server performing a step for tracking the at least one client usage of the one or more services during each session based on the received one or more metering packets in order to store data from which it can be determined whether the at least one client usage of the one or more services during each session is a charged time portion or a free time portion.

19. A computer program product as recited in claim 18, wherein a plurality of metering packets are received that include both a session-ending metering packet and a session-in-progress metering packet.

20. A computer program product as recited in claim 18, wherein the method further comprises a step for authenticating the received metering packets.

21. A computer program product as recited in claim 18, wherein a plurality of metering packets are received that are redundant, the method further comprising a step for discarding the received metering packets that are redundant.

22. A computer program product as recited in claim 18, wherein each metering packet further comprises (i) a packet type element, (ii) a sequence number element, (iii) a session identifier element, and (iv) a packet authentication element, and wherein the time element comprises a charged time portion corresponding to some access to the one or more services that incurs an access charge, and a free time portion corresponding to other access to the one or more services that does not incur an access charge.

23. A computer program product as recited in claim 18, wherein the method further comprises a step for communicating one or more usage tracking parameters to the at least one client, wherein the one or more usage tracking parameters include at least one of (i) an indication that the at least one client should track usage of the one or more services provided by the one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the at least one client should send metering packets.

24. In a computer network that comprises at least one server, the at least one server providing one or more services to at least one client that accesses the one or more services through the one or more servers during a plurality of sessions created in response to a login request from the at least one client, with at least some of the plurality of sessions occurring simultaneously, and wherein access to the one or more services during a particular session includes at least one of a charged time portion and a free time portion, a method of tracking the at least one client usage of the one or more services during each session, the method comprising acts of:

a client sending a login request to a login service; accessing, through one or more sessions created in response to the login request, at least one of the one or more services provided by one or more servers and tracking parameters corresponding to client usage of the one or more services;

generating a plurality of metering packets corresponding to a single session that each includes a time element indicating the at least one client usage of the one or more services, each metering packet being used at the client to store data for tracking usage of the one or more services during each session, and each metering packet comprising a data structure for storing the following data:

a session identifier element that links a particular metering packet with a particular session; and

15

a time element indicating the at least one client usage of the one or more services, the time element comprising a charged time portion and a free time portion, wherein the charged time portion corresponds to access to one or more services that incurs an access charge, and wherein the free time portion corresponding to access to one or more services that does not incur an access charge; and sending at least one of the plurality of metering packets to a census service, wherein the census service updates a usage database based on the metering packets so that the usage database reflects the at least one client usage of the one or more services provided by the at least one server.

25. A method as recited in claim 24, wherein a plurality of metering packets are generated and sent over regular, periodic intervals, and wherein the metering packets includes both a session-ending metering packet and a session-in-progress metering packet.

26. A method as recited in claim 24, further comprising acts of:

- receiving a session key associated with the one or more sessions;
- hashing at least a portion of each metering packet and the corresponding session key to generate an authentication element; and
- storing each authentication element in the corresponding metering packet.

27. A method as recited in claim 24, further comprising an act of sending redundant metering packets to the census service using a communication protocol that does not guarantee delivery.

28. A method as recited in claim 24, wherein the time element comprises a charged time portion corresponding to some access to the one or more services that incurs an access charge, and a free time portion corresponding to other access to the one or more services that does not incur an access charge.

29. A method as recited in claim 28, wherein each metering packet further comprises (i) a packet type element, (ii) a sequence number element, (iii) a session identifier element, and (iv) a packet authentication element.

30. A method as recited in claim 24, further comprising an act of receiving one or more headers from the login service, wherein the one or more headers include at least one of (i) an indication that the at least one client should track usage of the one or more services provided by the one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the at least one client should send metering packets.

16

31. A method as recited in claim 24, further comprising an act of storing metering information in non-volatile memory.

32. A method as recited in claim 31, further comprising an act of sending the stored metering information to the census service in a subsequent session.

33. A computer program product comprising:
a computer readable medium for carrying machine-executable instructions that implement the method of claim 24.

34. A computer program product as recited in claim 33, wherein a plurality of metering packets are generated and sent over regular, periodic intervals, and wherein metering packets include both a session-ending metering packet and a session-in-progress metering packet.

35. A computer program product as recited in claim 33, wherein the method further comprises acts of:

- receiving a session key associated with the one or more sessions;
- hashing at least a portion of each metering packet and the corresponding session key to generate an authentication element; and
- storing each authentication element in the corresponding metering packet.

36. A computer program product as recited in claim 33, wherein the method further comprises an act of sending redundant metering packets to the census service using a communication protocol that does not guarantee delivery.

37. A computer program product as recited in claim 33, wherein each metering packet further comprises (i) a packet type element, (ii) a sequence number element, (iii) a session identifier element, and (iv) a packet authentication element.

38. A computer program product as recited in claim 33, wherein the method further comprises an act of receiving one or more headers from the login service, wherein the one or more headers include at least one of (i) an indication that the at least one client should track usage of the one or more services provided by the one or more servers, (ii) a unique session identifier, and (iii) a metering interval indicating how frequently the at least one client should send metering packets.

39. A computer program product as recited in claim 33, the method further comprising an act of storing metering information in non-volatile memory.

40. A computer program product as recited in claim 39, wherein the method further comprises an act of sending the stored metering information to the census service in a subsequent session.

* * * * *

APPENDIX B-13



US006877007B1

(12) **United States Patent**
Hentzel et al.

(10) **Patent No.:** **US 6,877,007 B1**
(45) **Date of Patent:** **Apr. 5, 2005**

(54) **METHOD AND APPARATUS FOR TRACKING A USER'S INTERACTION WITH A RESOURCE SUPPLIED BY A SERVER COMPUTER**

2003/0037138 A1 * 2/2003 Brown et al. 709/225

OTHER PUBLICATIONS

IBM TDB, Algorithm to Inhibit Record/Play in a Non-Display Field, Apr. 1990, vol. 32, No. 11, PP. 462-465.*

* cited by examiner

Primary Examiner—Jack M Choules
(74) *Attorney, Agent, or Firm*—Christensen O'Connor Johnson Kindness PLLC

(76) Inventors: **Anna M. Hentzel**, 2218 McKinley Ct., Ames, IA (US) 50010; **Timothy I. Hentzel**, 440 Utah St., San Francisco, CA (US) 94110; **Robert R. Hentzel**, 440 Utah St., San Francisco, CA (US) 94110; **Brian F. Allen**, 19 Rausch, Apt. C., San Francisco, CA (US) 94103

(57) **ABSTRACT**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 302 days.

A method and system is provided for tracking a user's interaction with a resource or resources supplied by a server computer. When a user requests a Web page from a server for viewing using a Web browser, the user is redirected to a tracking server. The tracking server sends the requested Web page, with an embedded script to the user's computer along with an application program that can record the user's interaction with the Web page. As the user interacts with the Web page, input made by the user, such as mouse movements, button clicks, typing, etc. is streamed back to the tracking server by the application program. The recorded session may be later retrieved from the tracking server for playback. Based on the user's input, the tracked resource may be modified, such as making it more user friendly or more easily navigable.

(21) Appl. No.: **09/978,845**

(22) Filed: **Oct. 16, 2001**

(51) **Int. Cl.**⁷ **G06F 17/30**

(52) **U.S. Cl.** **707/10; 709/224**

(58) **Field of Search** **707/10; 709/224**

(56) **References Cited**

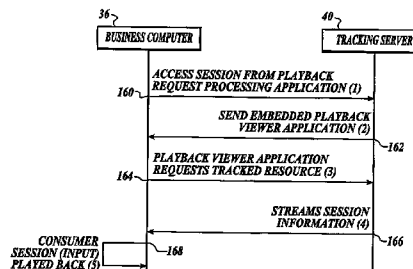
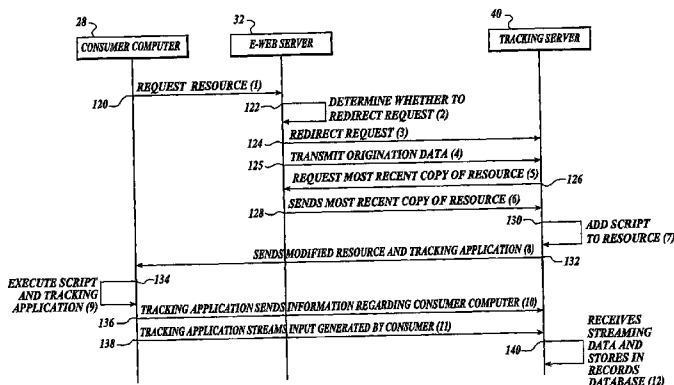
U.S. PATENT DOCUMENTS

5,675,510 A * 10/1997 Coffey et al. 709/224

5,796,952 A * 8/1998 Davis et al. 709/224

6,108,637 A * 8/2000 Blumenau 705/7

5 Claims, 13 Drawing Sheets



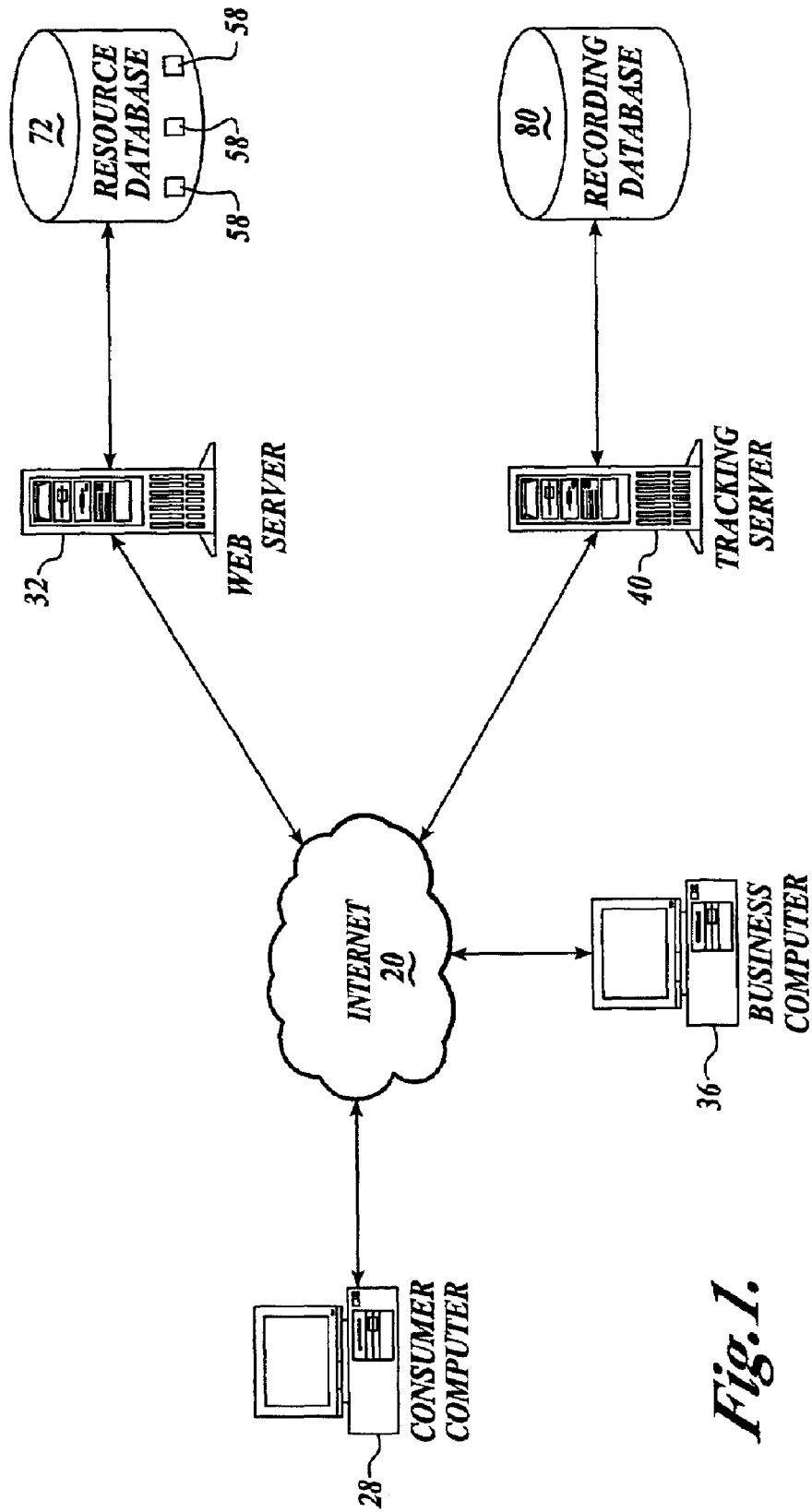


Fig. 1.

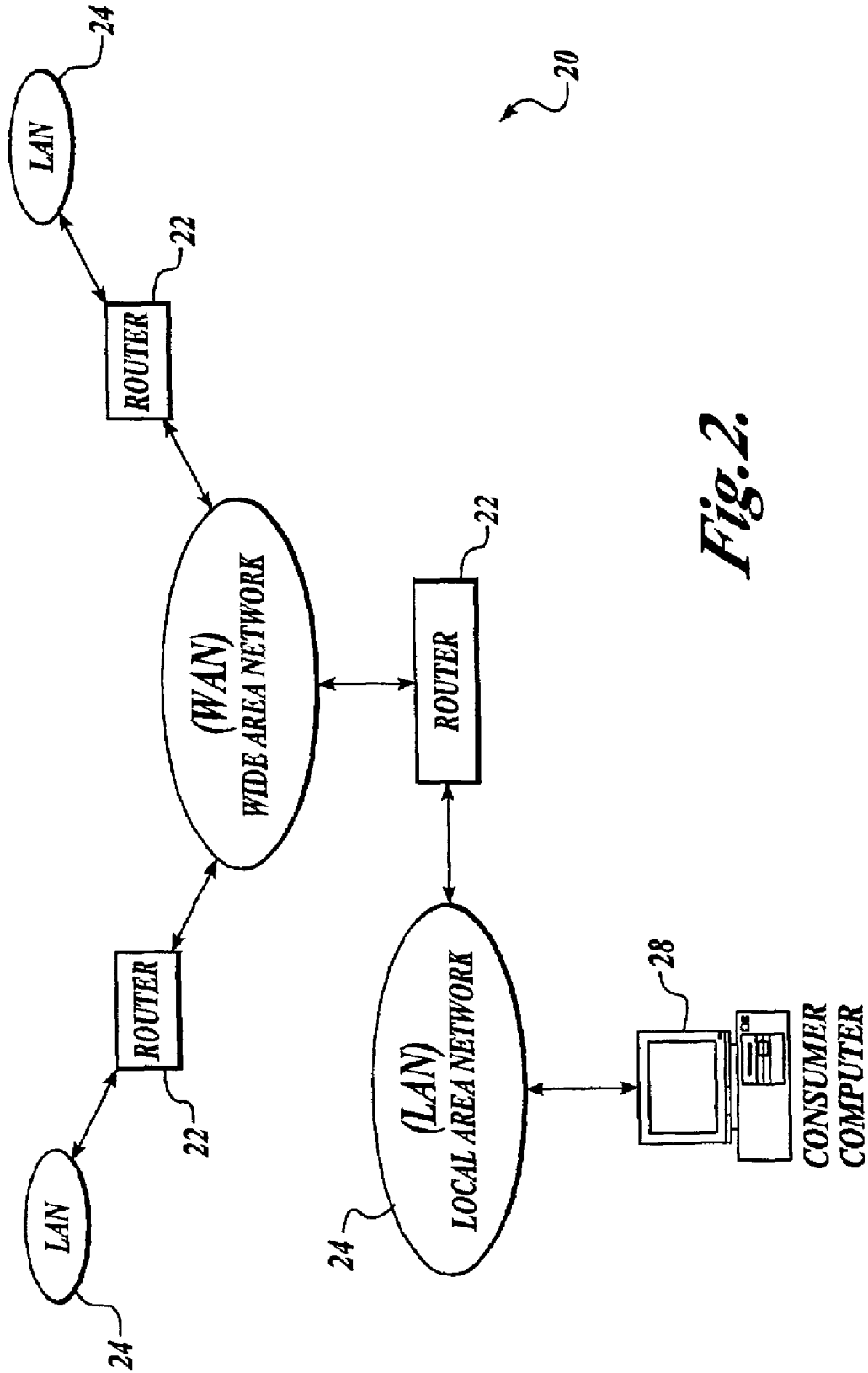


Fig. 2.

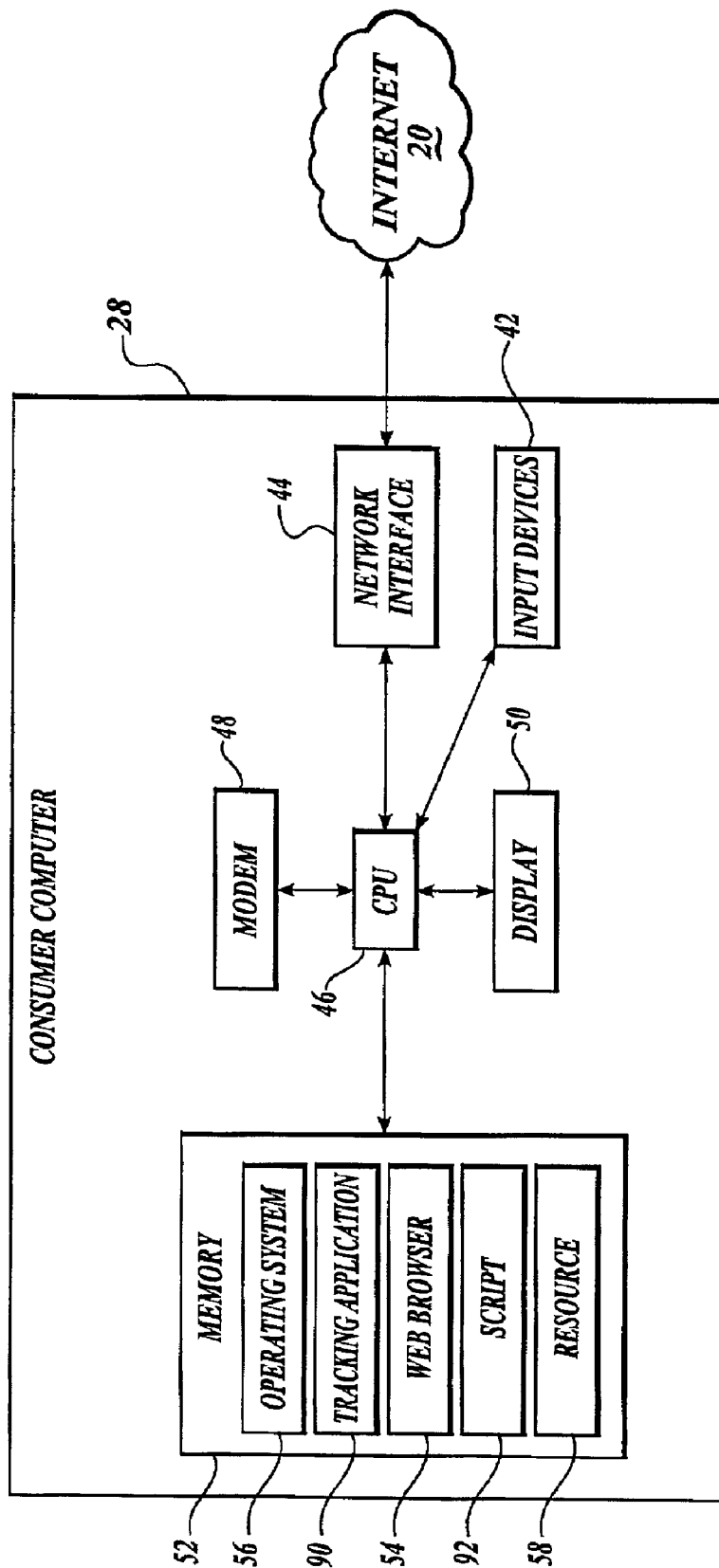


Fig. 3.

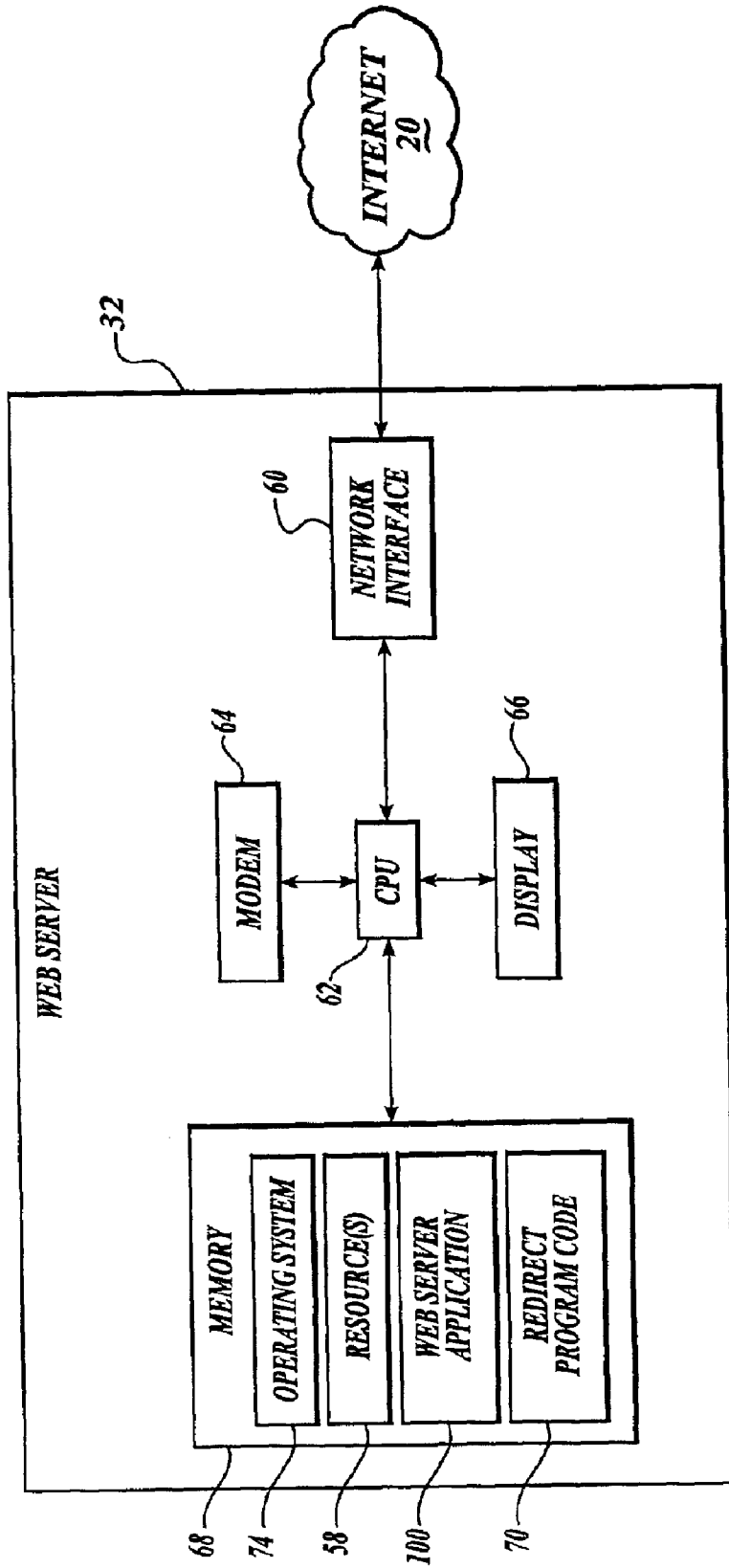


Fig. 4.

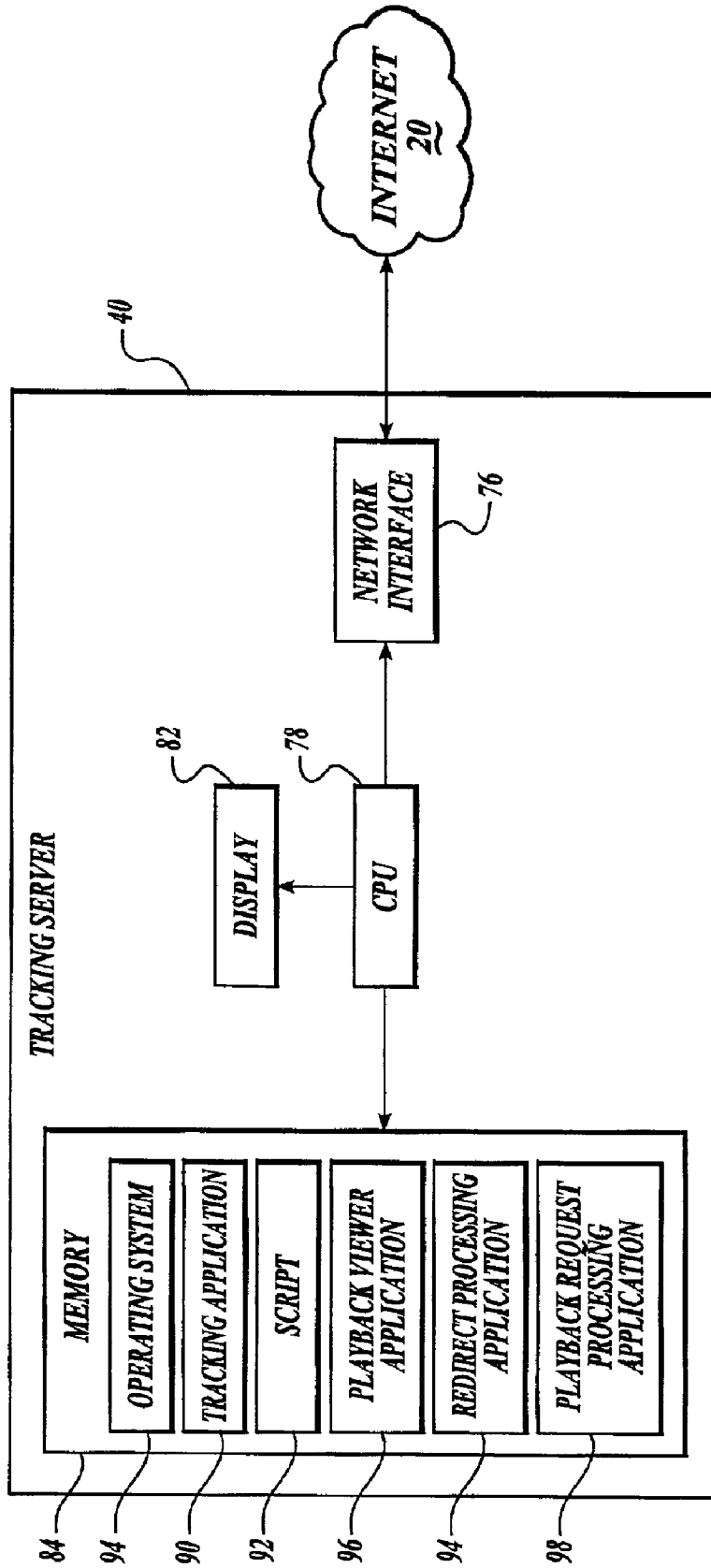


Fig. 5.

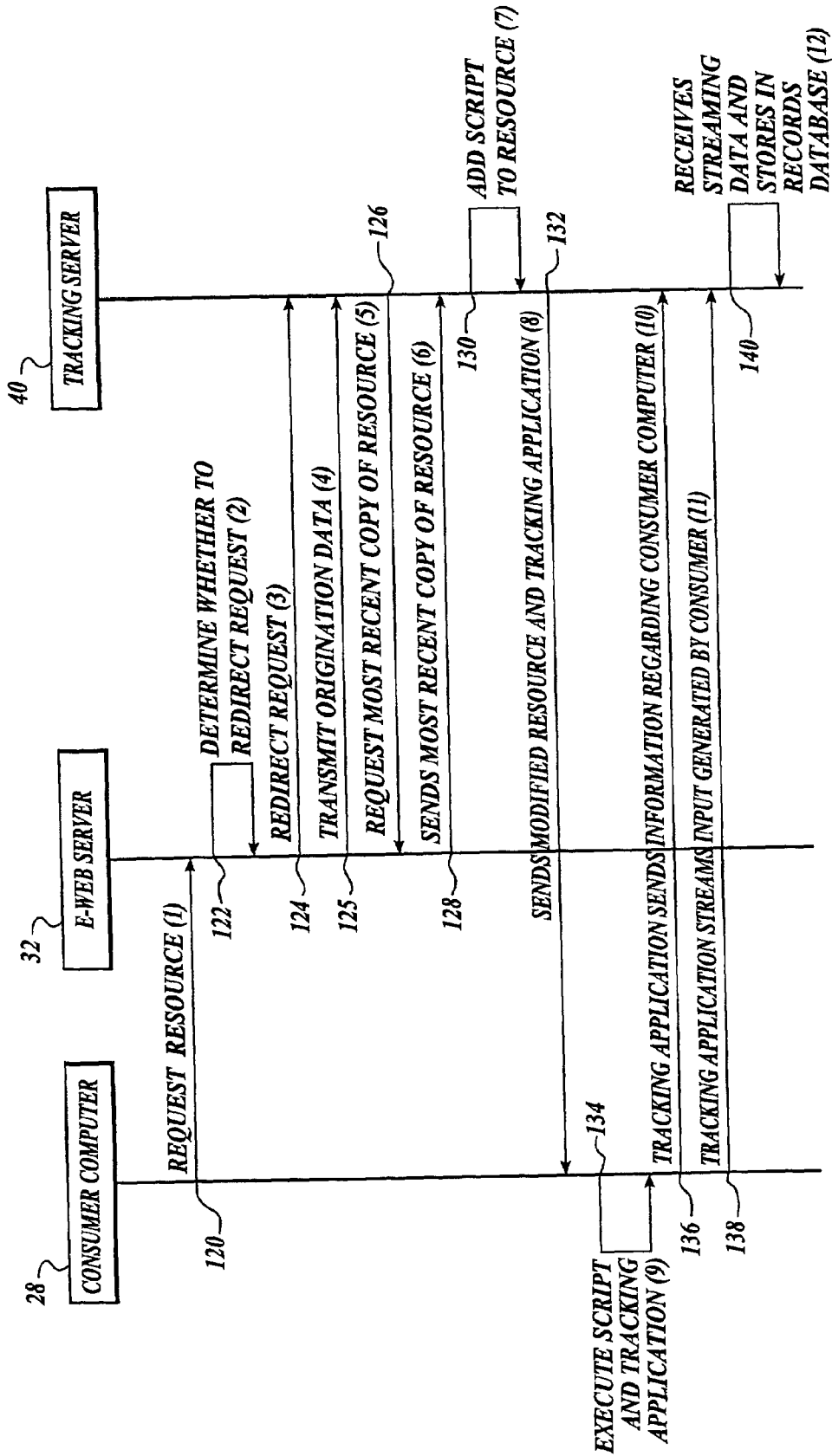


Fig. 6.

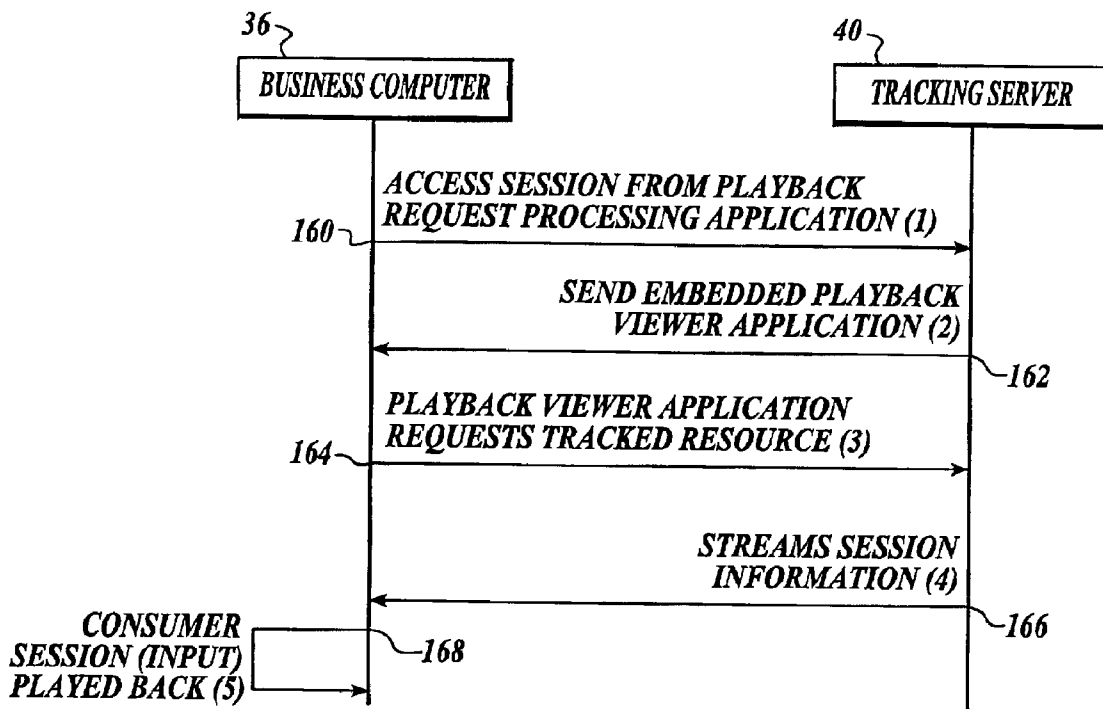


Fig. 7.

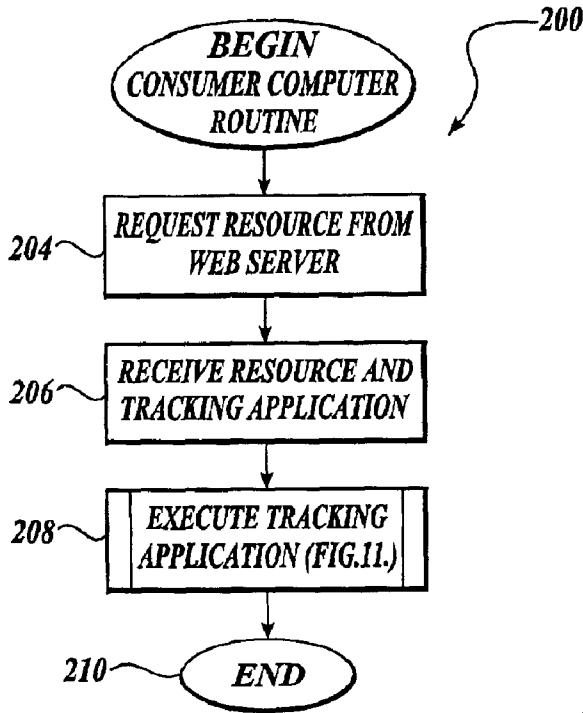


Fig. 8.

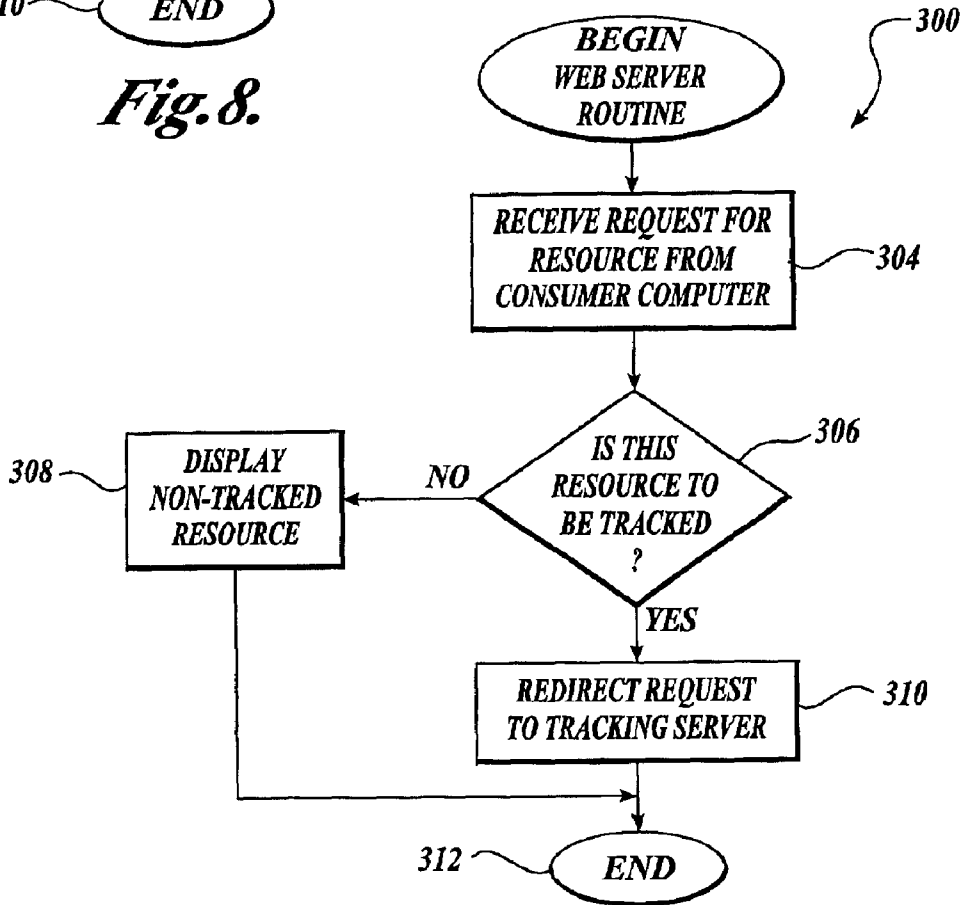


Fig. 9.

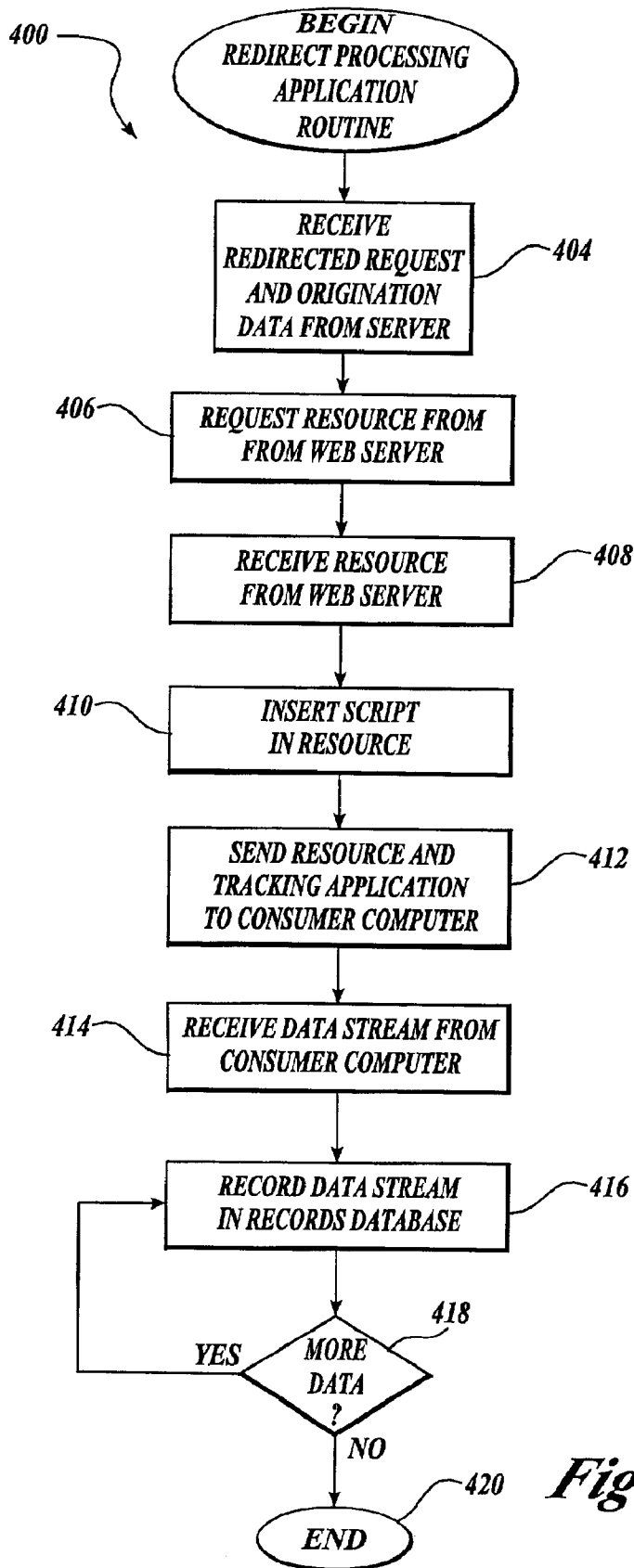


Fig. 10.

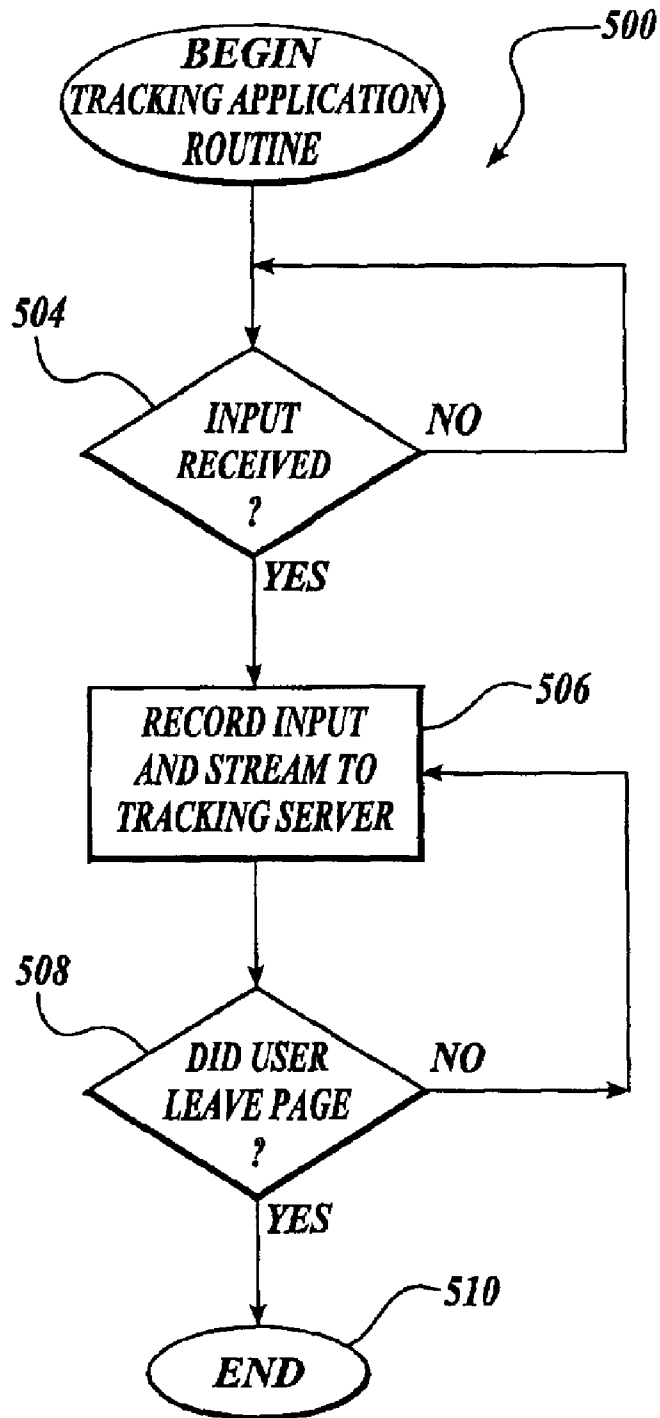


Fig. 11.

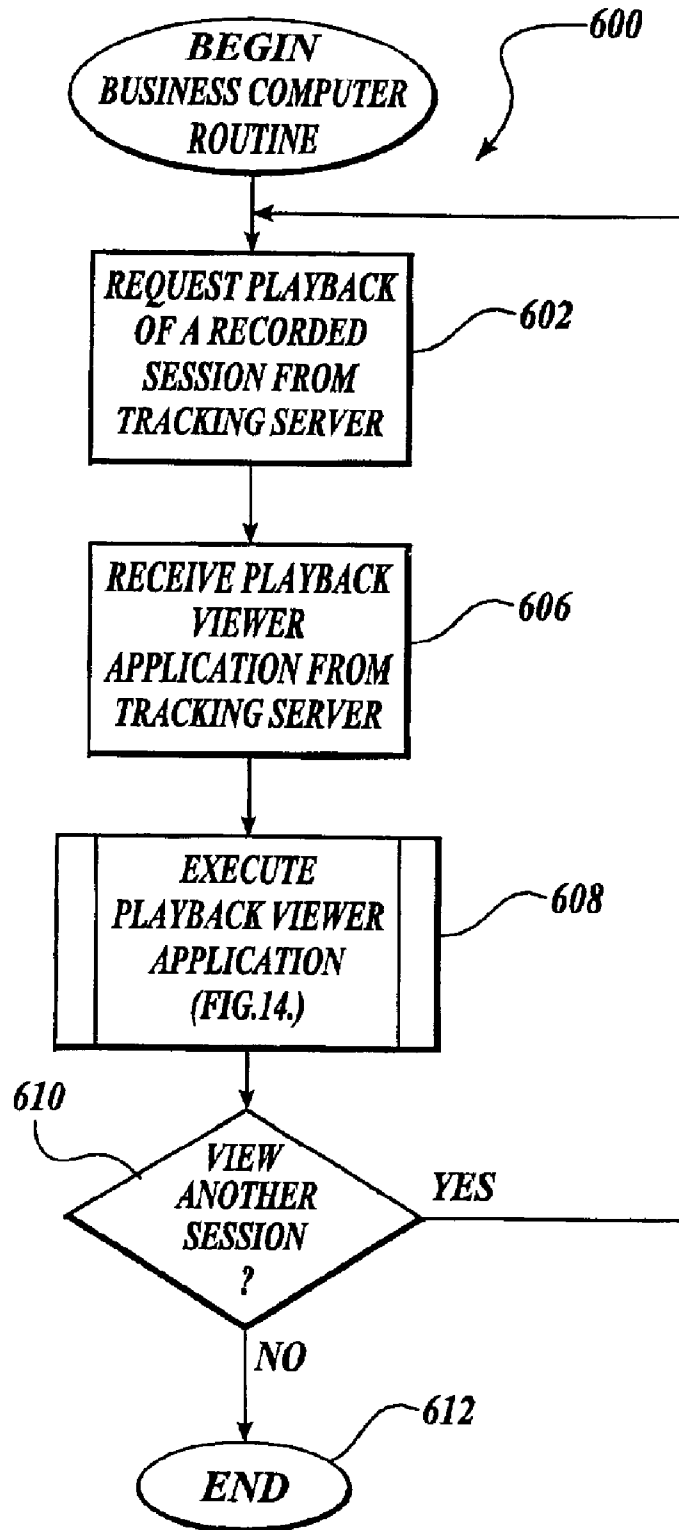


Fig.12.

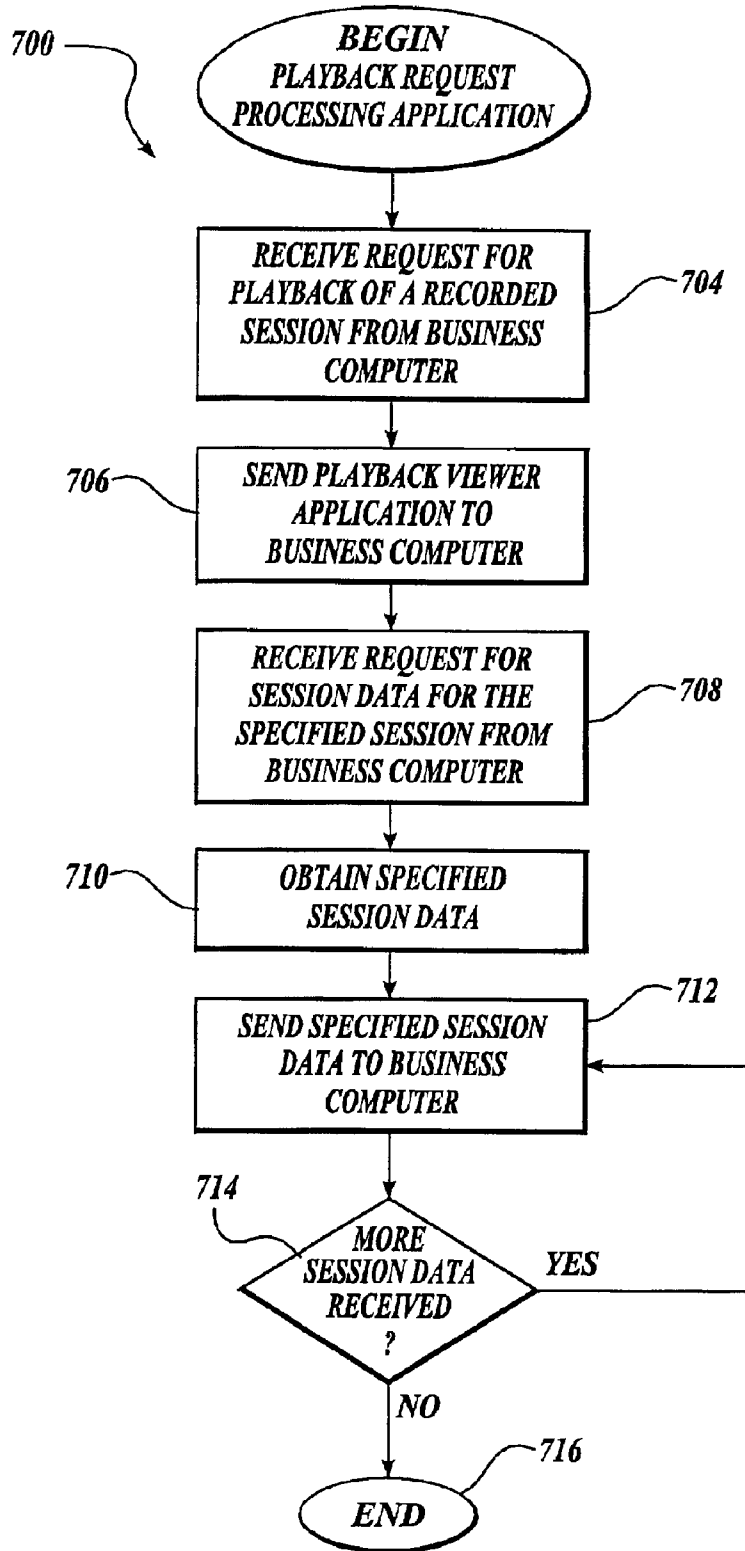


Fig. 13.

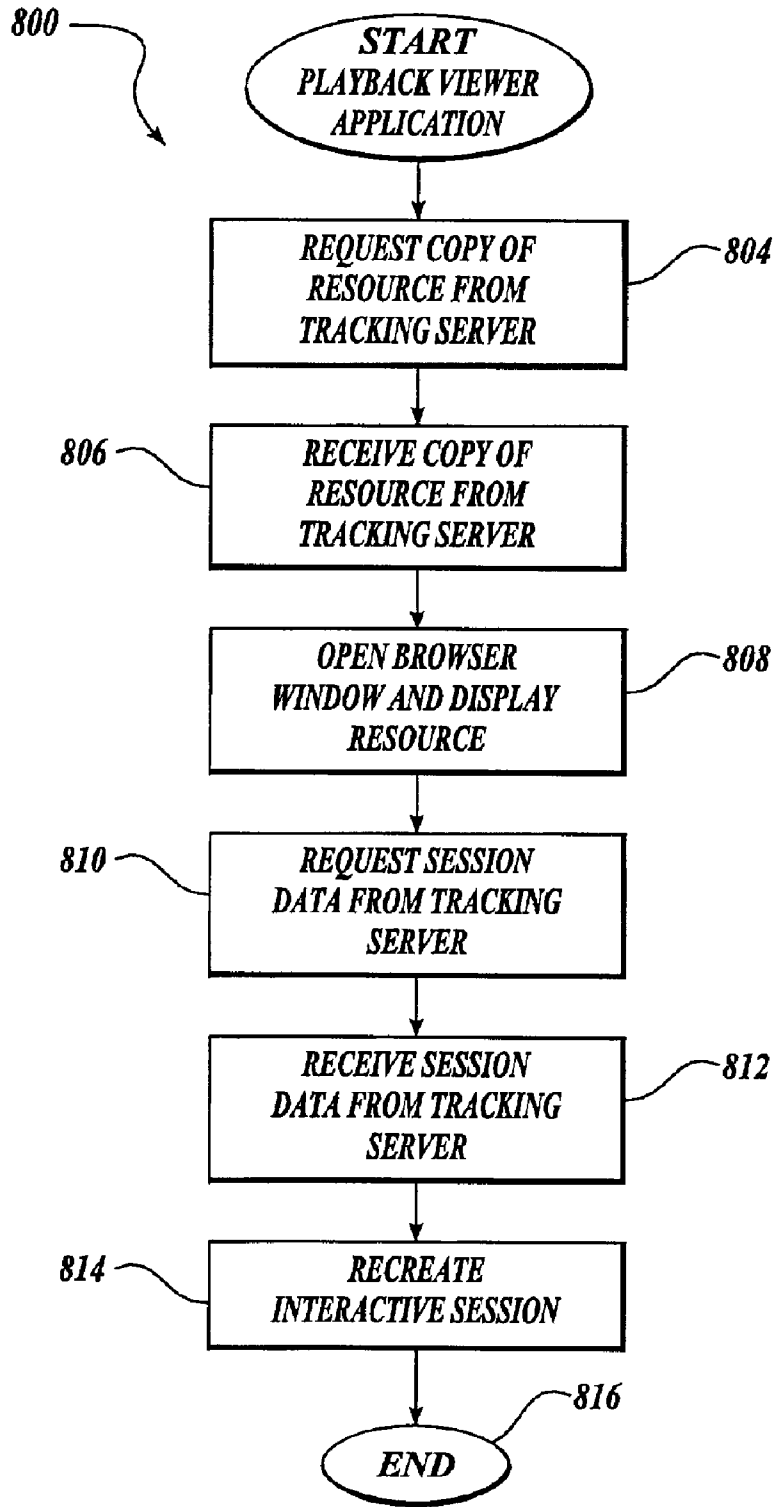


Fig. 14.

**METHOD AND APPARATUS FOR
TRACKING A USER'S INTERACTION WITH
A RESOURCE SUPPLIED BY A SERVER
COMPUTER**

FIELD OF THE INVENTION

In general, the present invention relates to computer software, and, in particular, to a method for tracking a user's interaction with a server computer.

BACKGROUND OF THE INVENTION

It is a truth universally acknowledged that, in order to succeed, a business must study the habits, desires, and behavior of its customers. For companies conducting business over the Internet and the World Wide Web ("Web" or "WWW"), this necessarily extends to examining and measuring their customers' interaction with their Web sites.

Commercial software currently exists to perform this analysis at the page navigation level. Such software allows companies to track and analyze such information as total traffic to a particular Web page, advertising revenue, and referral visits. However, current software tracking tools do not provide user interface ("UI") designers and marketing personnel the ability to study the users' interaction within a single Web page. Specifically, current software does not permit the analysis of how a single user interacts with a particular Web page. For example, a company may wish to know how a user interacts with the location of specific content of the Web page so that it can optimize its placement. Presently, for companies to receive this kind of feedback from a potential user, companies must conduct experiments with test subjects in controlled environments. Typically, a human observer is utilized to physically observe and record a user's interaction with a Web page. Experimental testing such as this is expensive and may lead to less than accurate results, which are unacceptable in today's business environment.

Therefore, in light of these deficiencies, there is a need for a method and apparatus for tracking a user's interaction with a single resource, such as a Web page, or multiple resources, such as Web sites. There is a further need for a method and system for tracking a user's interaction with a single or multiple resources that can provide an extremely accurate and inexpensive analysis of said interaction.

SUMMARY OF THE INVENTION

A method and apparatus for tracking a user's interaction with a resource supplied by a server computer is provided to overcome the deficiencies in the prior art. The present invention comprises a Web-based tool that allows every aspect of a user's interaction with a targeted Web page (or Web site with multiple Web pages) to be transparently recorded and played back. The transparency is important because it removes any analogues of the "Hawthorne Effect," that may be introduced during artificial usability studies overseen by humans. By developing a tool that can record and play back a user's interaction with specific Web pages, the business or resource designer of the Web pages can view and analyze the user's interaction for ways to improve the Web pages, such as making it more user friendly, more easily navigable, and more effective advertising.

In accordance with an aspect of the present invention, a method is provided for tracking a user's interaction with a

resource is provided. A server computer obtains a request for a resource. The server computer transmits the resource and a program module for capturing data describing the user's interaction with the resource to a client computer. A stream of data describing the user's interaction with the resource, such as mouse movement, mouse clicks, etc., is transmitted from the program module to the server computer and saved.

In accordance with another aspect of the present invention, a method is provided for tracking a user's interaction with a resource. In response to a request for a tracked resource stored at a server computer, the tracked resource and a program module for capturing data describing the user's interaction with the resource is received. The user's interaction with the resource is captured by the program module and is transmitted as a data stream.

In accordance with yet another aspect of the present invention, a method is provided for replaying a user's interactive session with a tracked resource. In particular, an application program is provided that may request data describing an interactive session from a server computer. Data representative of the specific recorded session is received by the application program along with a copy of the tracked resource. The application program utilizes the data describing the interactive session and the copy of the tracked resource to recreate and display the interactive session.

A computer-readable medium and a computer-controlled apparatus are also provided for tracking a user's interaction with a server computer.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIG. 1 is a block diagram showing an illustrative operating environment for implementing aspects of the present invention;

FIG. 2 is a block diagram showing a representative portion of the Internet;

FIG. 3 is a block diagram depicting an illustrative architecture for a consumer computer utilized to view and interact with a resource supplied by a server computer in accordance with an actual embodiment of the present invention;

FIG. 4 is a block diagram depicting an illustrative architecture for a server computer utilized to provide a resource to a consumer computer in accordance with aspects of the present invention;

FIG. 5 is a block diagram depicting an illustrative architecture for a tracking server utilized to record a user's interaction via a consumer computer with a resource from a server computer in accordance with an actual embodiment of the present invention;

FIG. 6 is a parallel functionality diagram depicting the interaction between a consumer computer, a Web server, and a tracking server in accordance with an actual embodiment of the present invention;

FIG. 7 is a parallel functionality diagram depicting the interaction between the business computer and the tracking server when replaying a recorded session in accordance with an actual embodiment of the present invention;

FIG. 8 is a flow diagram illustrating a routine implemented by the consumer computer for generating data describing an interactive session with a tracked resource in accordance with aspects of the present invention;

3

FIG. 9 is a flow diagram illustrating a routine implemented by a server computer for initiating the tracking of an interactive session in accordance with an actual embodiment of the present invention;

FIG. 10 is a flow diagram illustrating a routine implemented by a redirect processing application of a tracking server for processing the redirected consumer request in accordance with aspects of the present invention;

FIG. 11 depicts the execution of a tracking application routine performed by a consumer computer;

FIG. 12 is a flow diagram illustrating a routine implemented by a business computer for viewing a previously recorded interactive session generated by a tracking application in accordance with an actual embodiment of the present invention;

FIG. 13 is a flow diagram illustrating a routine implemented by a playback request processing application for processing a request to play back a previously recorded interactive session with respect to an actual embodiment of the present invention; and

FIG. 14 depicts the execution of a playback viewer application routine performed by a business computer, according to an actual embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In accordance with the present invention, an illustrative embodiment of a system for tracking a user's interaction with a resource, such as a Web page, is generally shown in FIG. 1. In operation, a computer user requests a resource 58, such as a Web page or other displayable data file, from a server computer, such as a Web server 32, for viewing using a Web browser executing on the consumer computer 28. The Web server 32 determines whether to redirect the user to a different server computer 40 (hereinafter referred to as "tracking server") that can record the user's interaction with the specific Web page of the Web server 32. The interaction by the user that may be recorded may include, but is not limited to, mouse movements, keyboard strokes, menu scrolling, etc. If the Web server 32 redirects the user to the tracking server 40, the tracking server 40 receives the redirected user request and transmitted origination data from the Web server 32.

Next, the tracking server requests and receives the most recent version of the Web page from the Web server 32 and merges the page retrieved from the Web server 32 with a client-side executable component that can initiate the execution of a program module for transmitting data describing the user's interaction with the Web page to the tracking server 40. In turn, the tracking server 40 sends the modified or merged resource, in this case the Web page, with the embedded script to the consumer computer 28 along with the program module (hereinafter referred to as "tracking application") that can record the user's interaction with the resource 58. As the user interacts with the resource, i.e. Web page, all of the inputs such as mouse movements, button clicks, typing, etc. generated by the user through various input devices are recorded by the tracking application and streamed back to the tracking server 40. At a later date, a business running the Web server 32 can access the recorded session from the tracking server 40 for analysis. Based on the recorded input, the business or resource designer can view and analyze the user's input to identify ways to improve the resource, such as making it more user friendly or more easily navigable.

Referring now to FIG. 2, an illustrative operating environment for an embodiment of the present invention will be

4

described. Aspects of the present invention are implemented as an executable software component located on a server computer, such as the tracking server 40, accessible via the Internet. As is well known to those skilled in the art, the term "Internet" refers to the collection of networks and routers that use the Transmission Control Protocol/Internet Protocol ("TCP/IP") to communicate with one another. A representative section of the Internet 20 is shown in FIG. 1, in which a plurality of local area networks ("LANs") 24 and a wide area network ("WAN") 26 are interconnected by routers 22. The routers 22 are special purpose computers used to interface one LAN or WAN to another. Communication links within the LANs may be twisted wire pair, or coaxial cable, while communication links between networks may utilize 56 Kbps analog telephone lines, 1 Mbps digital T-1 lines, 45 Mbps T-3 lines or other communications links known to those skilled in the art. Furthermore, a consumer computer 28 and other related electronic devices can be remotely connected to either the LANs 24 or the WAN 26 via a modem and temporary telephone or wireless link. It will be appreciated that the Internet 20 comprises a vast number of such interconnected networks, computers, and routers and that only a small, representative section of the Internet 20 is shown in FIG. 2.

The Internet has recently seen explosive growth by virtue of its ability to link computers located throughout the world. As the Internet has grown, so has the WWW. As is appreciated by those skilled in the art, the WWW is a vast collection of interconnected or "hypertext" documents written in HyperText Markup Language ("HTML"), or other markup languages, that are electronically stored at "WWW sites" or "Web sites" throughout the Internet. Other interactive hypertext environments may include proprietary environments such as those provided by America Online or other online service providers, as well as the "wireless web" provided by various wireless networking providers. One skilled in the relevant art will appreciate that the present invention can be implemented in any such interactive hypertext environments.

A WWW site is a server/computer connected to the Internet that has mass storage facilities for storing hypertext documents and other types of resources and that runs administrative software for handling requests for those stored hypertext documents. A hypertext document normally includes a number of hyperlinks, i.e., highlighted portions of text which link the document to another hypertext document possibly stored at a WWW site elsewhere on the Internet. Each hyperlink is associated with a Uniform Resource Locator ("URL") that provides the exact location of the linked document on a server connected to the Internet and describes the document. Thus, whenever a hypertext document is retrieved from any WWW server, the document is considered to be retrieved from the WWW. As is known to those skilled in the art, a WWW server may also include facilities for storing and transmitting application programs, such as application programs written in the JAVA® programming language from Sun Microsystems, for execution on a remote computer. Likewise, a WWW server may also include facilities for executing scripts and other application programs on the WWW server itself.

A consumer or other remote user may retrieve hypertext documents from the WWW via a WWW browser application program. A WWW browser, such as Netscape's NAVIGATOR® or Microsoft's Internet Explorer, is a software application program for providing a graphical user interface to the WWW. Upon request from the user via the WWW browser, the WWW browser accesses and retrieves the

5

desired hypertext document from the appropriate WWW server using the URL for the document and a protocol known as HyperText Transfer Protocol (“HTTP”). HTTP is a higher-level protocol than TCP/IP and is designed specifically for the requirements of the WWW. It is used on top of TCP/IP to transfer hypertext documents between servers and clients. The WWW browser may also retrieve application programs from the WWW server, such as JAVA applets, for execution on the consumer computer 28.

Referring back to FIG. 1, an actual embodiment of the present invention will now be described. A user or consumer computer 90 connects to the Internet 20 through a modem or other type of connection. Once connected to the Internet 20, a user of the consumer computer 28 may utilize a WWW browser to view and interact with Web pages on WWW sites such as a WWW site provided by the Web server 32. As is known to those skilled in the art, the consumer computer 28 may comprise a general purpose personal computer capable of executing a WWW browser. The consumer computer 28 may also comprise another type of computing device such as a palm-top computer, a cell phone, personal digital assistant, and the like. Consumer computer 28 is described in greater detail below with respect to FIG. 3.

Turning now to FIG. 3, an illustrative architecture for the consumer computer 28 utilized to view and interact with a resource 58 supplied by the Web server 32 will be described. Those of ordinary skill in the art will appreciate that the consumer computer 28 includes many more components than those shown in FIG. 3. However, it is not necessary that all of these generally conventional components be shown in order to disclose an illustrative embodiment for practicing the present invention.

As shown in FIG. 3, the consumer computer 28 includes a network interface 44 for connecting directly to a LAN or a WAN, or for connecting remotely to a LAN or WAN. Those of ordinary skill in the art will appreciate that the network interface 44 includes the necessary circuitry for such a connection, and is also constructed for use with the TCP/IP protocol, the particular network configuration of the LAN or WAN it is connecting to, and a particular type of coupling medium. The consumer computer 28 may also be equipped with a modem 48 for connecting to the Internet through a point to point protocol (“PPP”) connection or a SLIP connection as known to those skilled in the art.

The consumer computer 28 also includes a processing unit 46, a display 50, and a memory 52. The memory 52 generally comprises a random access memory (“RAM”), a read-only memory (“ROM”) and a permanent mass storage device, such as a disk drive. The memory 52 stores an operating system 56 for controlling the operation of the consumer computer 28. In one actual embodiment of the invention, the operating system 56 provides a graphical operating environment, such as Microsoft Corporation’s WINDOWS® graphical operating system in which activated application programs are represented as one or more graphical application windows with a display visible to the user.

The memory 52 also includes a WWW browser 54, such as Netscape’s NAVIGATOR® or Microsoft’s Internet Explorer browser, for accessing the WWW. It will be appreciated that these components may be stored on a computer-readable medium and loaded into the memory 52 of the consumer computer 28 using a drive mechanism associated with the computer-readable medium, such as a floppy, CD-ROM or DVD-ROM drive. The memory 52 may also include a tracking application 90, resources 58, and a

6

script 92 received from the tracking server 40 via the Internet 20. As will be described in greater detail below, the user’s interaction with the resources (through the use of the WWW browser 54) may be recorded by the tracking application 90.

The memory 52, network interface 44, display 50, and modem 48 are all connected to the processing unit 46 via one or more buses. Consumer computer 28 may also include several input devices 42 such as pointing devices, keyboards, or light pens which are connected to the processing unit 46 via one or more buses. As would be generally understood, other peripherals may also be connected to the processing unit in a similar manner.

As mentioned briefly above, a server computer generally designated as Web server 32 is also connected to the Internet 20. The Web server 32 comprises a general purpose server computer and is described in more detail below with reference to FIG. 4. The Web server 32 stores resources, such as Web pages, and receives requests for resources from the consumer computer 28. For instance, a user operating the consumer computer 28 may wish to receive information regarding flight information from a travel Web site running on the Web server 32. In response to these requests, the Web server 32 determines whether to redirect the request from the consumer computer 28 to the tracking server 40. If the Web server 32 determines that the user’s request should be redirected, the Web server 32 redirects the request to the tracking server 40 so that the user’s interaction with the requested resource may be recorded. According to an embodiment of the present invention, only a portion of those users requesting resources from the Web server 32 are redirected to the tracking server 40.

Referring now to FIG. 4, an illustrative architecture for a Web server 32 utilized to provide resources 58 to the consumer computer 28 will be described. As used herein the term “resource” comprises any type of data file or data files that a user may view or otherwise interact with utilizing a Web browser, such as HTML. Those of ordinary skill in the art will appreciate that the Web server 32 includes many more components than those shown in FIG. 4. However, it is not necessary that all of these generally conventional components be shown in order to disclose an illustrative embodiment for practicing the present invention. Moreover, although the computer system described in FIG. 4 is described as a server, it will be appreciated that the function of the document server may be implemented by computer systems not generally classified as server-type computer systems. Further, although only one Web server 32 is depicted in FIG. 1, it will be appreciated that other Web servers 32 may be located elsewhere on the Internet 20 and be utilized to serve resources 58 to a consumer computer 28.

As shown in FIG. 4, the Web server 32 includes a network interface 60 for connecting directly to a LAN or a WAN, or for connecting remotely to a LAN or WAN. Those of ordinary skill in the art will appreciate that the network interface 60 includes the necessary circuitry for such a connection, and is also constructed for use with the TCP/IP protocol, the particular network configuration of the LAN or WAN it is connecting to, and a particular type of coupling medium.

The Web server 32 also includes a processing unit 62, a display 66, and a mass memory 68. The mass memory 68 generally comprises a RAM, a ROM and a permanent mass storage device, such as a hard disk drive, tape drive, optical drive, floppy disk drive, or combination thereof. The memory 68 stores an operating system 74 for controlling the

operation of the Web server **32**. It will be appreciated that the operating system component **74** may comprise a general-purpose server operating system as is known to those of ordinary skill in the art, such as UNIX, LINUX™, or Microsoft WINDOWS NT®.

The memory **68** may include one or more resources **58** which are to be provided in response to requests from WWW browsers. The Web server application **100** receives and responds to such requests. As will be described below, each resource **58** to be tracked contains a redirect program code **70** generated by the playback request processing application of the tracking server that will redirect a portion of the requests from the consumer computer to the tracking server. Along with redirecting the request, the redirect program code **70** transmits origination data to the tracking server, as will be described in more detail below. Furthermore, the resource **58** may be retrieved from a database **72** (FIG. 1). It will be appreciated that these components may be stored on a computer-readable medium and loaded into memory **68** of the Web server **32** using a drive mechanism associated with the computer-readable medium, such as a floppy, CD-ROM or DVD-ROM drive. The memory **68**, network interface **60**, display **66**, and modem **64** are all connected to the processing unit **62** via one or more buses. As would be generally understood, other peripherals may also be connected to the processing unit in a similar manner.

A business computer **36** is also connected to the Internet **20** and may be utilized to control the operation of the Web server **32**. The business computer **36** may also comprise a general purpose computer capable of executing a WWW browser program. The business computer **36** is maintained and operated by a business **38** wanting to conduct business over the Internet **20**. The business computer **36** is utilized to create, customize, maintain, and operate an e-commerce WWW site ("Web site") on the Web server **40**. For example, the business **38** may want to establish a travel business on the Internet and may use business computer **36** to create the Web site located on the Web server **40**. The business computer **36** may include components similar to the consumer computer **28** described in greater detail above with respect to FIG. 3.

A tracking server **40** is also connected to the Internet **20**, and may be utilized by the business **38** to record and playback a user's interactive session with the resource(s) **58**. As used herein, the term "session" or "interactive session" may comprise all inputs made by a user via input devices such as a mouse, keyboard or the like, when viewing and interacting with a Web site, which can be a single tracked resource or multiple tracked resources **58**. These inputs can also be referred to as data or events. While the tracking server **40** is described and illustrated herein as being a remote server, separate from the Web server **32**, one skilled in the relevant art will appreciate that the Web server **32** may execute the functions of both the Web server **32** and the tracking server **40** described herein.

Referring now to FIG. 5, an illustrative architecture for the tracking server **40** will be described. Those of ordinary skill in the art will appreciate that the tracking server **40** includes many more components than those shown in FIG. 5. However, it is not necessary that all of these generally conventional components be shown in order to disclose an illustrative embodiment for practicing the present invention.

As shown in FIG. 5, the tracking server **40** is connected to the Internet **20** via a network interface **76**. Those of ordinary skill in the art will appreciate that the network interface **76** includes the necessary circuitry for connecting

the tracking server **40** to the Internet **20**, and is constructed for use with the TCP/IP protocol, the particular network configuration of the LAN or WAN it is connecting to, and a particular type of coupling medium.

The tracking server **40** also includes a processing unit **78**, a display **82**, and a mass memory **84**, all connected via a communication bus, or other communication device. The mass memory **84** generally comprises a RAM, ROM, and a permanent mass storage device, such as a hard disk drive, tape drive, optical drive, floppy disk drive, or combination thereof. The mass memory **84** stores an operating system **94** for controlling the operation of the tracking server **40**. It will be appreciated that this component may comprise a general-purpose server operating system as is known to those of ordinary skill in the art, such as UNIX, LINUX™, or Microsoft WINDOWS NT®.

The mass memory **84** also stores a redirect processing application **94** and the tracking application program **90**. The tracking application **90** comprises computer executable instructions which, when executed by the consumer computer **28**, causes the consumer computer **28** to stream data back to the tracking server **40** that describes the user's interactive session. The tracking application will be described in greater detail below with respect to FIG. 11.

Mass memory **84** also stores a script **92** which is sent along with the tracking application **90** via the Internet **20** to the consumer computer **28**. The script **92** contains code that causes the consumer computer **28** to execute the tracking application **90**. The tracking application **90** and script **92** are sent with a copy of the requested resource **58** retrieved from the Web server **32** by redirect processing application **94**. As mentioned above and described in detail below, the tracking application **90** and script **92** operate to cause of stream of data describing the user's interactive session to be sent to the tracking server **40**.

The mass memory **84** also stores a playback viewer application **96** and a playback request processing application **98** for allowing business **38** to replay the recorded session. The operation of the playback viewer application **96** and the playback request processing application **98** will be described in greater detail with respect to FIGS. 13 and 14. It will be appreciated that all of the above-described components may be stored on a computer-readable medium and loaded into the memory **84** of the tracking server **40** using a drive mechanism associated with the computer-readable medium, such as a floppy, CD-ROM or DVD-ROM drive.

The tracking server **40** is also operatively connected to a recording database **80** (FIG. 1) for storing data describing the interactive sessions between a user and a tracked resource. For instance, the tracking server **40** may store the following data in the recording database **80** for each interactive session: user location, basic information about the consumer computer, such as platform, resolution, color depth, browser version, and clipboard data, inputs such as keystrokes, mousemoves, mouseclicks, and scrolling that affects the browser window provided by the operator of the consumer computer using computer input devices such as a keyboard, mouse or light pen, and the state of various controls on the resource (Web page), e.g. the fact that a checkbox is selected or not.

Interactive sessions may be grouped by location, i.e. URL, and may be sorted by discrete data, such as click-stream (sort by session), visitor, date, or length of session. One skilled in the relevant art will appreciate that the sessions may be sorted by other data or criteria as well. The business **38** may also assign an additional classification (or

“type”) to the session (e.g. “new user”) for future analysis. Those skilled in the art should appreciate that the recording database 80 may be stored locally on tracking server 40, or remotely at other computers in a networked computing environment like the Internet 20.

The recording of interactive sessions between a user and the resource 58 from the Web server 32 will now be described in detail with respect to FIGS. 6 and 7. The following description describes the recording of interactive sessions with a signal resource for illustrative purposes only, and therefore should not be construed as limiting. One skilled in the art will appreciate that the interactive session with multiple resources may also be recorded. Referring to FIG. 6, a user requests a tracked resource 58 or Web page from the Web server 32 utilizing a Web browser at block 120. The business 38, prior to a user requesting the resource 58, determines the resources 58 or Web pages of the Web site that should be tracked. When a resource 58 to be tracked is requested at block 120, the Web server 32 determines whether or not to track the resource at block 122. If the requested resource is to be tracked, the Web server 32 redirects the request to the tracking server at block 124. Along with redirecting the request, the Web server 32 transmits origination data, such as a HTTP header, to the tracking server 40 at block 125. In an actual embodiment of the present invention, the Web server 32 may randomly redirect a percentage of the users requesting the resource to the tracking server 40. Additionally, once the user is being tracked, the user can be automatically tracked as they move from page to page throughout the web site.

The redirect processing application 94 running on the tracking server 40 receives the redirected user request and the origination data from the Web server 32. Next, at block 126, the processing application 94 requests a copy of the most recent version of the resource 58 requested by the user, from the Web server 32. The redirect processing application 94 utilizes the origination data so that the tracking server receives the exact resource which can accurately mimic the original request. After the Web server 32 sends the requested resource at block 128, the processing application 94, at block 130, merges the resource retrieved from the Web server 32 with the script 92 and changes the base tag and modifies any script and framesets contained within the resource to reference the Web server 32. At block 132, the processing application 94 sends the proxied resource, merged with the script 92, along with the tracking application 90 to the consumer computer 28. The tracking server permanently stores a copy of the requested resource.

When the tracked resource is loaded by the Web browser executing on the consumer computer 28, the script 92 and the tracking application 90 are executed at block 134. The script 92 retrieves information about the consumer computer 28 such as browser type, screen resolution, and the like, and sends this information to the tracking application 90. The tracking application 90 then opens a connection to the redirect processing application 94 on the tracking server 40 and sends the information about the consumer computer at block 136. The redirect processing application 94 formats this session information and inserts it into the recording database 80 via a Java Database Connectivity (“JDBC”) application program interface (API). This database entry associates the business 38, the location of the resource, the session data, and the visitor with the data stream that is about to be sent by the tracking application 90.

At block 138, the tracking application 90 streams data describing the user’s input with the resource, such as typing and scrolling, to the tracking application 90, so that it can be

forwarded to and stored at the associated database 80 at block 140. At block 140, the tracking application 90 batches the events, compresses the data, and securely streams them back to the redirect processing application 94 on the tracking server 40 via the connection. The database 80 maintains a persistent storage of the sessions, their related events, and any relevant user information such as location.

Referring now to FIG. 7, a functionality diagram will be described illustrating how a recorded session may be viewed by the business 38 operating Web server 32. When the business 38 decides to analyze a user’s interaction with one of the tracked resources 58, a representative of the business 38 will access the recorded session from the playback request processing application 98 on tracking server 40 at block 160. At block 162, the tracking server 40 sends the playback viewer application 96 to the business computer 36 to be executed. Each of the sessions recorded by the tracking server 40 is available via the playback request processing application 98. Sessions are grouped by location and may be sorted by data such as visitor, date, length of session and the like. At this point the business 38 may assign an additional classification (or “type”) to the session (e.g. “new user”) for future analysis.

The playback of a session is controlled by the playback viewer application 96. At block 164, the playback viewer application 96 requests the tracked resource 58 from the tracking server. The playback viewer application 96 launches another browser window or re-uses a currently open browser window at the business computer 36 to display the tracked resource exactly as it was shown to the user, including browser size and screen placement. After the browser window is obtained, the playback viewer 96 application requests the session data from the playback request processing application 98 on tracking server 40. At block 166, the playback request processing application 98 sends the session data to the business computer 36 to be viewed by the business 38. The session is then replayed in real time at block 168. The playback viewer application 96 uses the application program interface (API), such as Windows API or the like, to recreate the events as accurately as possible so that the browser will be unable to differentiate between a replayed mouseclick and an actual user click.

Referring now to FIG. 8, routine 200 will be described showing the operation of the consumer computer 28 when generating an interactive session. The routine 200 begins at block 204, where the user operating the consumer computer 28 requests the resource 58 from the Web server 32 using a Web browser. For example, the user may wish to receive flight information from a travel Web site operating on Web server 32. The user enters the specific URL for the Web site within the browser. This causes the browser to request the resource, i.e. Web page, associated with the identified URL.

After the user requests the resource from Web server 32 at block 204, the routine 200 proceeds to block 206, where the consumer computer 28 obtains or receives a copy or proxy of the requested resource 58 from the tracking server 40. Received along with the copy of the resource 58 is the tracking application 90 and script 92. As described above, the tracking application 90 can be an executable program or applet, as known in the art. It will be appreciated that the consumer computer may receive a message (visible or invisible) that indicates that the user request was redirected to the tracking server 40. In a preferred embodiment of the present invention, the user is not aware that her request has been redirected and has received an executable program that will record her interaction with the requested resource.

Next, at block 208, the tracking application 90 and script 92 are executed on the consumer computer 28. As will be

11

discussed in more detail below with respect to FIG. 11, the tracking application 90 receives data representing the user's input from input devices such as the keyboard or mouse associated with the interactive session between the user and the copy of the resource received from tracking server 40, and streams the associated data back to the tracking server 40 for storage. If the user leaves the tracked resource, the routine ends at block 210.

Referring now to FIG. 9, an illustrative routine 300 will be described that illustrates the operation of the Web server 32. The routine 300 begins at block 304, where the Web server 32 receives a request for a resource 58 from the consumer computer 28. For example, the travel Web site located on Web server 32 could receive a request from the consumer computer 28 for a resource containing flight information. After the request for the resource 58 is received, the logic proceeds to a decision block 306, where a determination is made as to whether the resource 58 is to be tracked.

In an actual embodiment of the present invention, the business 38 operating the Web server 32 decides which resource, i.e. Web page, is going to be tracked from the Web site. To this end, the business places an executable component, redirect program code 70, into the source code of every resource that is to be tracked. The playback request processing application 98, which will be described in more detail below, includes an "Add Tracker" feature that requires the business to input the URL, application server type, and specify the fraction of traffic that should be recorded. The tracking server 40 stores this information in an entry in a location table of the database 80. The redirect program code 70 is generated by a program module of the playback request processing application 98, such as an account manager program module, located on the tracking server 40 based on the business input of the "Add Tracker" feature. Those of ordinary skill in the art will appreciate that the playback request processing application 98 may include many more program modules that when executed, perform functions such as generating tracking statistics.

The small code block for each location, in this case, Web server 32 depends on the type of the server (e.g. Active Server Pages ("ASP"), Java Server Pages ("JSP"), standard HTML, etc.) and the fraction of traffic that is to be recorded. If, at block 306, it is determined that the requested resource is to be tracked, the routine 300 proceeds to block 310 where the Web server 32 redirects the request from the consumer computer 28, along with the transmission of origination data to the tracking server 40. In one embodiment, the origination data are HTTP headers that are stored in a hash map so that HTTP headers can be passed back to accurately mimic the original request. The routine then ends at block 312.

Returning to block 306, if the requested resource is not to be tracked, the routine proceeds to block 308, where the non-tracked resource is displayed. The routine ends at block 312.

Referring now to FIG. 10, a routine 400 illustrating the operation of the redirect processing application 94 will be described. The routine 400 begins at block 404, where the tracking server 40 receives the redirected request for the resource 58 and the origination data from the Web server 32. In one embodiment, as described above, the origination data in the form of HTTP headers is store in the hash map of the tracker server 40. From block 404, the routine 400 continues to block 406, where the tracking server 40 transmits a request to the Web server 32 for a copy or proxy of the resource 58 to be tracked. In an actual embodiment, the

12

redirect processing application 94 executing routine 400 opens an HTTP connection with the Web server 32 and utilizes the origination data to retrieve the most recent version of the resource, i.e. web page, requested by the user. For example, the redirect processing application 94 sends a request to retrieve the latest version of the travel web page that the user wishes to obtain flight information from.

From block 406, the routine 400 continues to block 408, where the redirect processing application 94 receives the resource, i.e. web page, from the Web server 32 and permanently stores it at the tracking server 40. The routine 400 then proceeds to block 410 where the redirect processing application 94 inserts the script 92 into the resource retrieved from the Web server 32 and changes the base tags and modifies any script and framesets to reference the Web server 32. After the script 92 has been inserted into the resource at block 410, the routine 400 continues to block 412, where the tracking server 40 sends the modified resource and the script 92 along with the tracking application 90 to the consumer computer 28.

From block 412, the routine 400 continues to block 414, where the redirect processing application 94 receives the data stream corresponding to the interactive session between the user and the resource. For example, as the user navigates (i.e. mouse movements and scrolling) through the travel Web page, each input event or interaction is streamed back to the tracking server 40 by the tracking application 90. The routine 400 proceeds to block 416 where the data stream is recorded and stored in the recording database 80. In an actual embodiment of the present invention, the redirect processing application 94 formats the interactive session information and inserts it into the records database 86 via JDBC. This insertion links the business 38, resource, and user with the transmitted stream of data. From this point on, every interactive input or event generated by the user, such as typing, mouse movements and scrolling, is sent to the tracking application 90 (as well as the browser) so that it can be forwarded to the database 90 at the tracking server 40. The tracking application 90 batches the events, compresses the data, and securely streams them back to the processing application 94 on the tracking server 40 via a HTTP connection.

After the redirect processing application 94 receives the data stream from the consumer computer 28, the logic proceeds to a decision block 418 and determines if more data is being received. As described above, if the user decides to leave the tracked page, the tracking application ceases to stream back data associated with the inputs of the user after the user has left the tracked resource. If so, the routine 400 returns to block 416 to continue to record the data stream in the records database 86. Otherwise, the Routine 400 ends at block 420.

As described briefly above with respect to FIG. 8, FIG. 11 depicts the execution of the tracking application in greater detail. Turning now to FIG. 11, such a routine 500 will now be described. The routine 500 begins at block 504 where a determination is made as to whether an input signal (e.g. mouse movement, keyboard stroke) has been received. If so, the routine 500 proceeds to block 506 where the tracking application records the input and streams the input to the tracking server 40. If no input is detected at block 504, the routine 500 returns to block 504 to continue to wait for an input to be received.

At block 506, the tracking application 90 records the input generated by the user while interacting with the resource. The tracking application 90 opens a connection between the

13

consumer computer **28** and the tracking server **40** and streams the input data back to the tracking server **40** to be stored. At decision block **508**, a determination is made as to whether the user has left the tracked resource. For example, the user could have “clicked” on a hyperlinked document within the resource, or typed in a new URL in the browser window. In either case, the user would leave the tracked resource and receive the resource that corresponds with the specific URL that was enter from either typing or “clicking” the mouse. In one embodiment of the present invention as described above, the redirect program code is operable to remain tracking the user throughout the Web site, regardless of whether the next requested resource has the redirect program code. The redirect program code would terminate tracking a resource after the user had left the Web site.

It will be appreciated that the user could request another resource that is to be tracked, either by Web server **32** or another business on the Internet **20** wanting their resource to be tracked. If this is the case, a new session is to be recorded and the routine **500** returns to block **202** to begin a new tracked session. If the routine determines that the user has left the tracked resource, the tracking application **90** returns at block **510** to the routine **200** shown in FIG. **8**. Otherwise, the routine returns to block **506** to continuously record and stream the input data from consumer computer **28** until the user leaves the resource in a way, for example, as described above. The tracking application **90** returns to block **510** if the user has left the tracked resource, where the logic of routine **200** proceeds to block **210** to end the routine.

Referring now to FIG. **12**, an illustrative routine **600** will be described for viewing a pre-recorded interactive session generated by the tracking application **90**. The routine **600** begins at block **604**, where the business **38** operating the business computer **36** requests playback of a recorded session from the tracking server **40**. For example, the business may wish to view a session generated between a user and a Web page of the travel Web site, or between a user on a web site located on the Web server **32**.

After the business **38** requests a session playback from the tracking server **40** at block **604**, the routine **600** proceeds to block **606** where the business computer **36** obtains or receives the playback viewer application **96**, such as an ActiveX control program, from the tracking server **40**. The routine **600** then continues to block **608**, where the playback viewer application **96** is executed on the business computer **36**. As will be discussed in more detail below, the playback viewer application **96** receives the stream data from the tracking server **40** and recreates the interactive session for review. Operation of the playback viewer application **96** is described in greater detail below with respect of FIG. **14**. The routine **600** proceeds from block **608** to decision block **610** where a determination is made as to whether the business **38** wants to view another session from a resource. If so, the routine returns to block **604**. Otherwise, the routine ends at block **612**.

Turning now FIG. **14**, an illustrative routine **800** will be described showing the operation of the playback viewer application. The routine **800** begins at block **804**, where the playback viewer application **96** requests a copy of the resource **58** from the tracking server **40**. At block **806**, the playback viewer application **94** receives a copy of the resource from the tracking server **40**. At block **808**, the playback viewer application **96** opens a browser window on the business computer **36**. One skilled in the relevant art will appreciate that the browser window may be new, or may be a previously used browser window. The browser window has the same dimensions as the browser originally utilized

14

by the user to display the resource. At block **810**, the playback viewer application **96** requests the session data from the tracking server **40** corresponding to the specific session to be viewed. At block **812**, the playback viewer application **96** receives the session data streamed from the tracking server **40**. In an actual embodiment of the present invention, the tracking server **40** includes a playback request processing application **98** that receives the request for session data, retrieves the session data from the associated database, and sends the session data to the playback viewer application **96**.

After receiving the session data at block **812**, the playback viewer application **96** recreates the interactive session by displaying the resource in the browser window and displaying the session data so as to recreate the user’s original interactive session with the resource. In an actual embodiment, the playback viewer application uses the application program interface (API) to recreate the session as accurately as possible so that the session substantially resembles the actual interactive session between the user and the tracked resource. For example, a copy of the travel Web page is opened into a browser window. The data representing the interactive session is played over the travel Web page to display the exact user interaction with the Web page. In this manner, the user’s original mouse movements, mouse clicks, screen scrolls, etc. will be replayed on the business computer in exactly the same manner as they were originally made by the user on the consumer computer. After playback of the session, the routine **800** ends at block **816**. It is to be understood that the playback viewer application **96** can have features such as pause, rewind, fast forward, so that the business **38** can properly analyze how the user interacted with the resource **58**. This analysis may then be used by the business to improve the resource in ways such as better design layout.

Referring now to FIG. **13**, an illustrative routine **700** will be described showing the operation of the playback request processing application **98**. The routine **700** begins at block **704**, where the playback request processing application **98** receives a request for a playback of the recorded session. In an actual embodiment, the recorded sessions are identified by the processing application **98** using parameters such as user, date, and length of session. Each parameter corresponds to the session data stored in database **86**. After the request for a playback of a session is received at block **704**, the routine **700** proceeds to block **706** where the processing application **98** sends a playback viewer application **96** to the business computer **36**. For example, a request can be made by “clicking” on the displayed parameter associated with the session the business wishes to review.

After the playback viewer application **96** is executed at the business computer **36** as described above with reference to FIG. **14**, the playback request processing application **98** receives, at block **708**, a request for session data associated with the session specified by the business computer at block **704**. Next, at block **710**, the playback request processing application **98** obtains the specified session data and sends the specified session data to business computer **36** at block **712**. It will be appreciated that the playback request processing application **98** can obtain the session data from the records database **80** or may obtain the data from the tracking server memory. The routine **700** then proceeds to the decision block **714** where a determination is made as to whether more session data is being obtained from, in this case, the records database. If so, the routine **700** returns to block **712** to continue sending the data to the business computer **36**. Otherwise, the routine **700** ends at block **716**.

15

The previously described implementation of the present invention provides advantages over software currently available in the art. The present invention provides a Web-based tool that allows every aspect of a user's interaction with a targeted Web site to be transparently recorded. By transparently recording the interaction, any analogues of the "Hawthorne Effect" that may be introduced during artificial usability studies overseen by humans are removed. By utilizing a tool that can record a user's interaction with a specific Web page or Web pages, a business or resource designer of the Web pages can view and analyze the user's input for ways to improve the Web pages, such as making it more user friendly or more easily navigable.

While the illustrative embodiment described above described a system and method for recording the interactive session between a user and a resource, such as a Web page, it will be readily evident to one skilled in the art that the system and method may record the interactive session between a user and multiple resources, such as a Web site. For example, in one embodiment, the redirect program code is operable to remain tracking the user throughout the Web site, regardless of whether the next requested resource has the redirect program code. In this instance, the session data recorded will represent one constant stream of data corresponding to the users interaction with the Web site (multiple resources).

While an illustrative embodiment of the invention has been illustrated and described, it will be appreciated that

16

various changes can be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A method for replaying a user's interactive session with a tracked resource, comprising:

- requesting data describing said interactive session;
- receiving said data and a copy of said tracked resource;
- displaying said tracked resource; and
- displaying said data describing said interactive session in a manner so as to recreate exactly said user's interactive session.

2. The method of claim 1, further comprising: receiving a program module for requesting said data representative of said interactive session and said copy of said tracked resource.

3. The method of claim 2, wherein said program displays said tracked resource and said data describing said interactive session.

4. A computer controlled apparatus capable of performing the method of any one of claims 1-3.

5. A computer readable medium comprising computer readable instructions which, when executed by a computer, cause the computer to perform the method of any one of claims 1-3.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,877,007 B1
DATED : April 5, 2005
INVENTOR(S) : A.M. Hentzel et al.


Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [56], **References Cited**, OTHER PUBLICATIONS, "IBM TDB," reference,
"Non-Display" should read -- Non-Display --.

Signed and Sealed this

Twenty-second Day of November, 2005

A handwritten signature in black ink on a light gray dotted background. The signature reads "Jon W. Dudas" in a cursive style.

JON W. DUDAS
Director of the United States Patent and Trademark Office

APPENDIX B-14



US007310609B2

(12) **United States Patent**
Middleton, III et al.

(10) **Patent No.:** **US 7,310,609 B2**
(45) **Date of Patent:** **Dec. 18, 2007**

(54) **TRACKING USER MICRO-INTERACTIONS WITH WEB PAGE ADVERTISING**

(75) Inventors: **Thomas M. Middleton, III**, Hingham, MA (US); **Gregory T. White**, Bedford, MA (US)

(73) Assignee: **Unicast Communications Corporation**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 784 days.

(21) Appl. No.: **10/100,631**

(22) Filed: **Mar. 14, 2002**

(65) **Prior Publication Data**

US 2002/0111865 A1 Aug. 15, 2002

Related U.S. Application Data

(63) Continuation of application No. 09/146,012, filed on Sep. 2, 1998, now Pat. No. 6,393,407.

(60) Provisional application No. 60/058,655, filed on Sep. 11, 1997.

(51) **Int. Cl.**
G06Q 40/00 (2006.01)

(52) **U.S. Cl.** **705/14; 705/27; 709/224**

(58) **Field of Classification Search** **705/14, 705/10, 26, 27, 40; 709/224, 226**
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,058,185 A * 10/1991 Morris et al. 382/305
5,434,863 A * 7/1995 Onishi et al. 370/402

5,491,820 A * 2/1996 Belove et al. 707/3
5,517,620 A * 5/1996 Hashimoto et al. 709/242
5,557,790 A * 9/1996 Bingham et al. 707/101
5,644,713 A * 7/1997 Makishima 709/242

(Continued)

FOREIGN PATENT DOCUMENTS

DE WO 9826346 * 6/1998

OTHER PUBLICATIONS

Traffic Cops For Web Servers, (Software vendors are offering tools to help network managers handle system resources being used by external & Internet traffic), Information Week, n 597, p. 44, Sep. 16, 1996.*

(Continued)

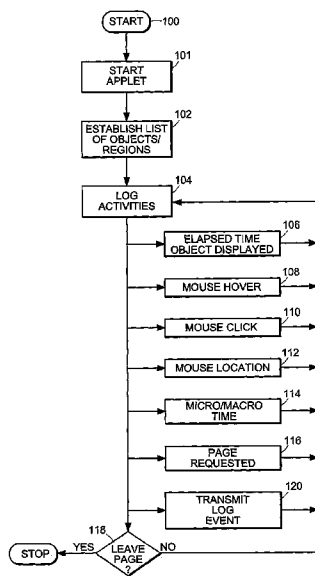
Primary Examiner—Hani M. Kazimi

(74) *Attorney, Agent, or Firm*—Michaelson & Associates; Peter L. Michaelson

(57) **ABSTRACT**

In connection with display of advertising within Web pages, an applet is downloaded to the user's Web browser to track the user's interactions with the Web page. Tracked user interactions include mouse cursor position, time displayed on page, time of mouse cursor hovering over the advertisement, and so on. At an appropriate time, such as when the display of the Web page is terminated, the applet forwards logged interaction information from the client to a remote server, the remote server being typically controlled by an advertiser, rating service or the like. As a result, the advertiser may track consumer response to advertising impressions on a Web page without requiring the user to download other pages. This allows advertisers to track user response to specific elements of the Web page as well as to better infer information about the user's interests in an effort to qualify the user prior to presenting subsequent advertising.

34 Claims, 3 Drawing Sheets



US 7,310,609 B2

Page 2

U.S. PATENT DOCUMENTS

5,649,185 A * 7/1997 Antognini et al. 707/9
5,675,741 A * 10/1997 Aggarwal et al. 709/242
5,694,545 A * 12/1997 Roskowski et al. 709/231
5,742,610 A * 4/1998 Natarajan 370/472
5,748,613 A * 5/1998 Kilk et al. 370/231
5,796,952 A * 8/1998 Davis et al. 709/224
5,848,397 A * 12/1998 Marsh et al. 705/14

5,999,914 A * 12/1999 Blinn et al. 705/26
6,054,984 A * 4/2000 Alexander 715/771

OTHER PUBLICATIONS

Internet: Ichat, Now Acuity, Launches Web Call Center, Network
briefing, PN/A, Trade, 1360-1369, Jun. 1, 1998.*

* cited by examiner

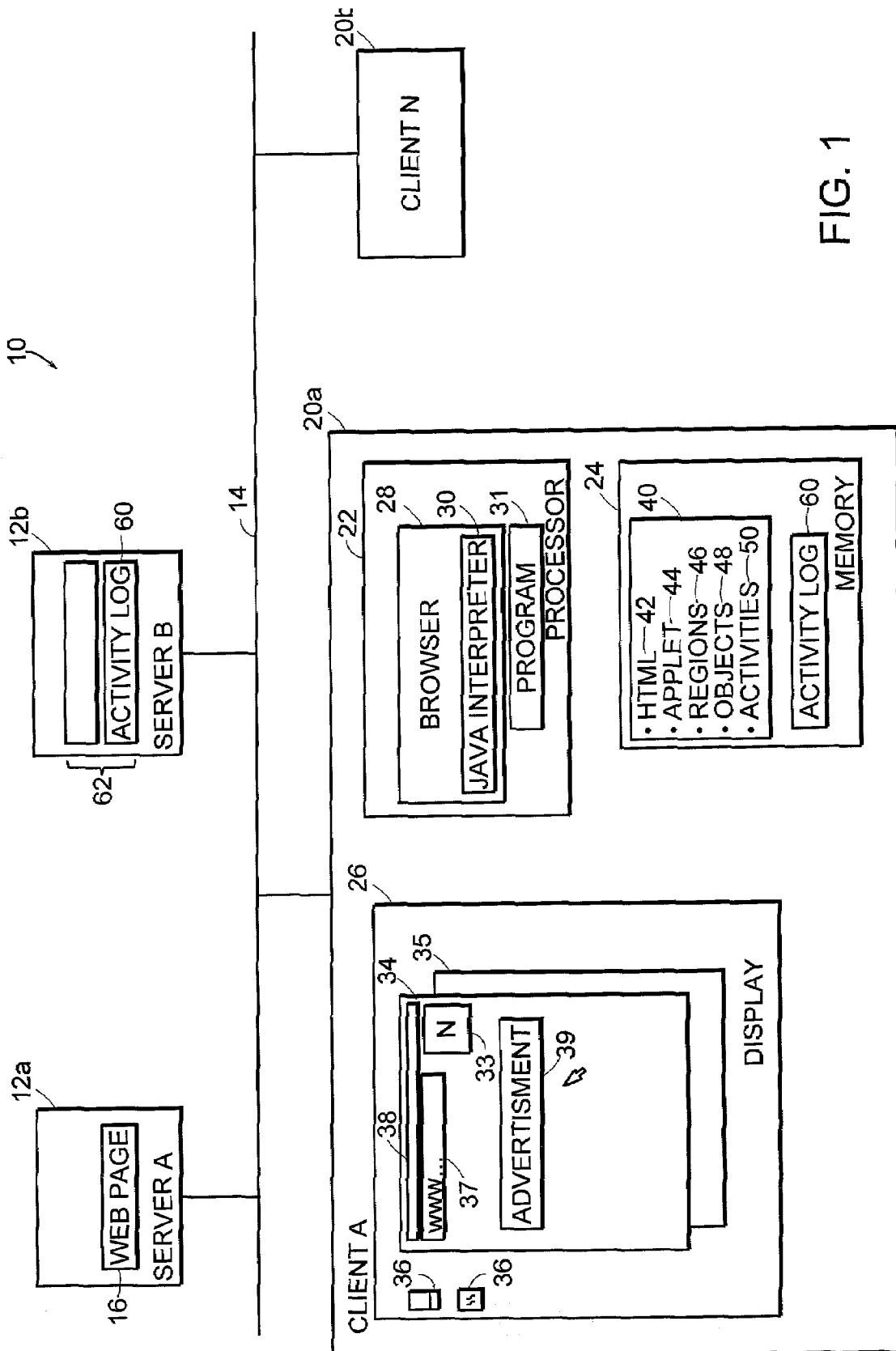


FIG. 1

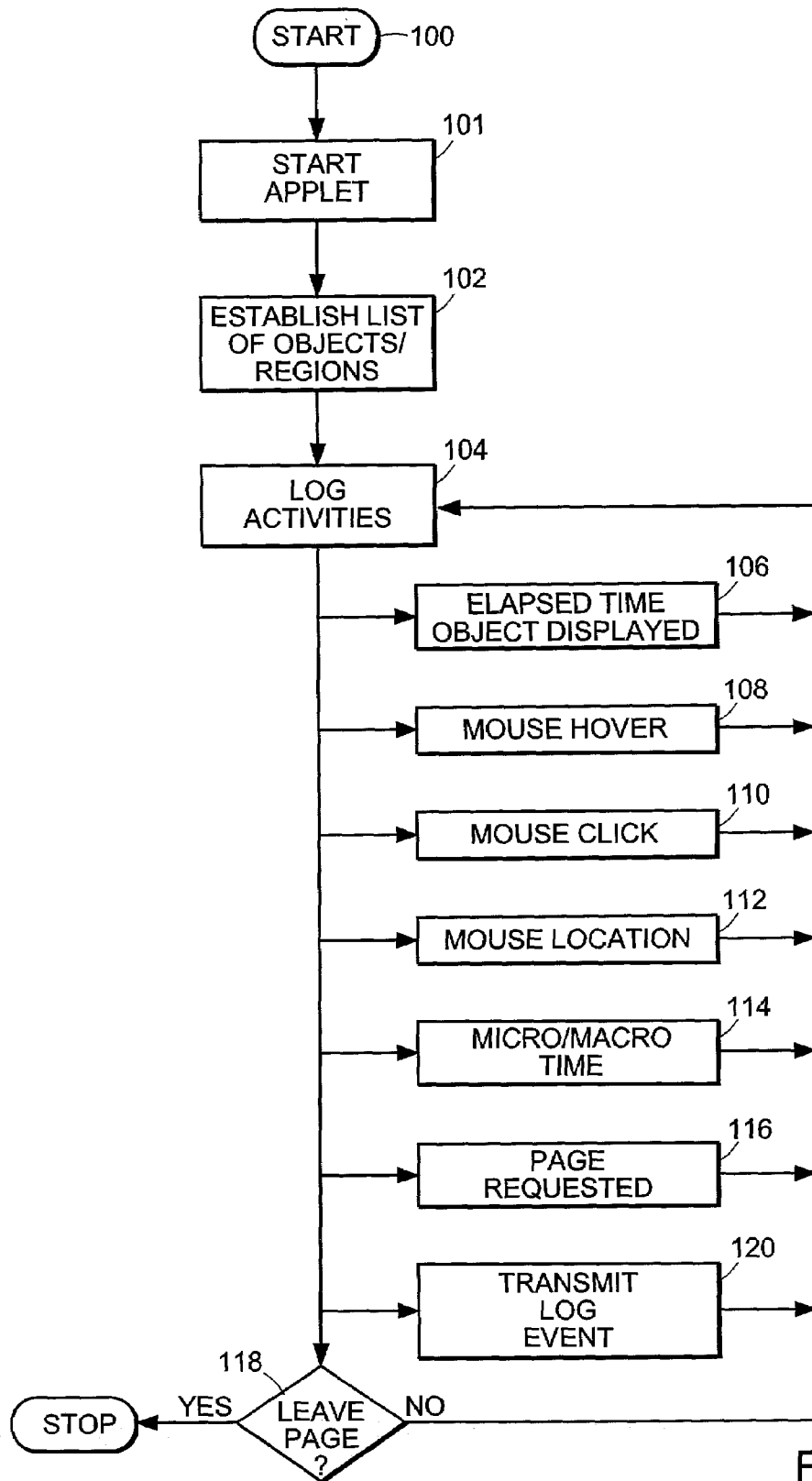


FIG. 2

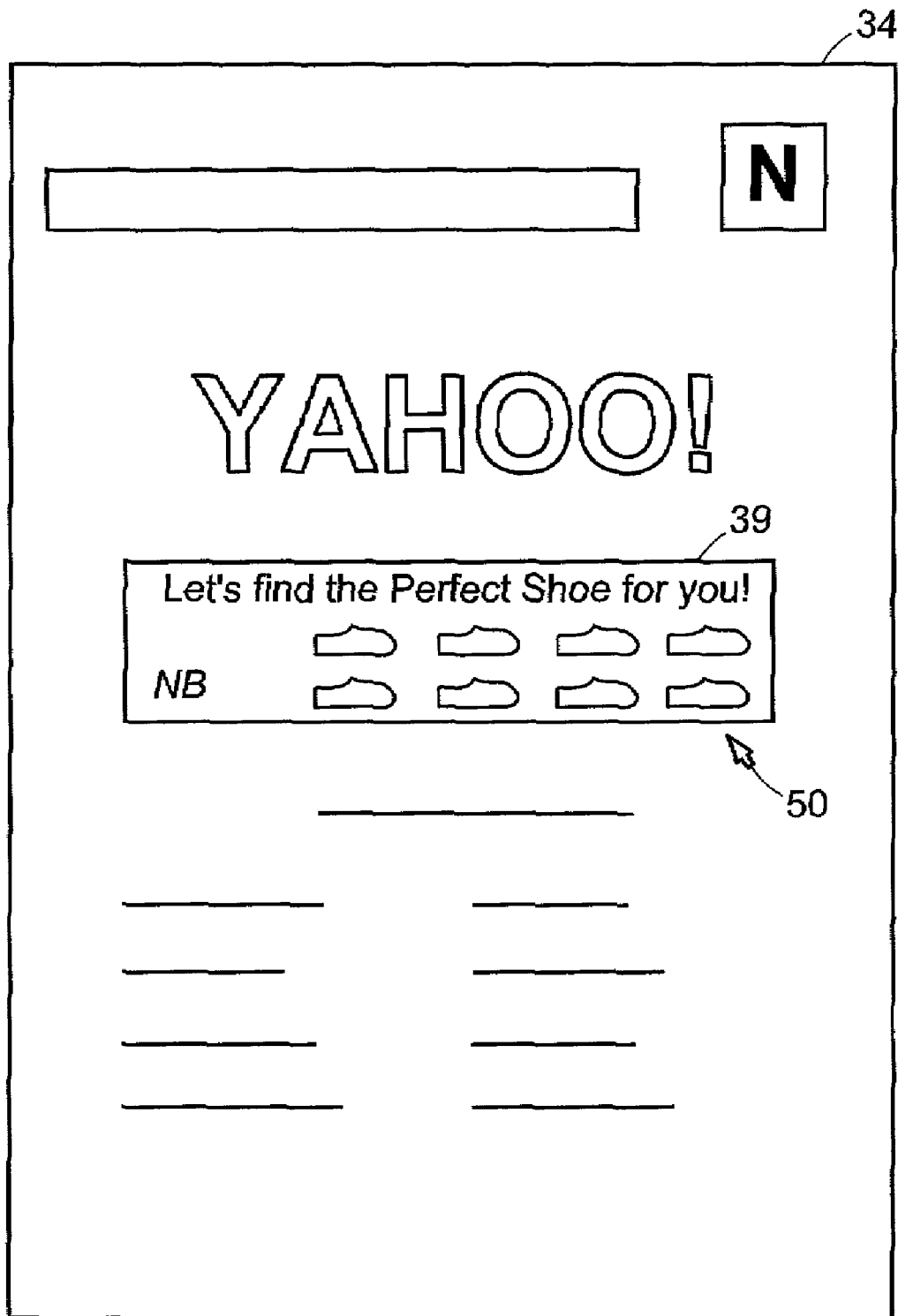


FIG. 3

TRACKING USER MICRO-INTERACTIONS WITH WEB PAGE ADVERTISING

RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 09/146,012, filed Sep. 2, 1998 which claims the benefit of U.S. Provisional Application No. 60/058,655, filed on Sep. 11, 1997.

The entire teachings of the above applications are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Distributed computing environments are becoming a very popular mechanism for publishing information of various types. In such an environment, a network of several different types of computers is used in order to share access to information. Certain computers, known as servers, contain databases and other repositories of information. Other computers in the network, known as clients, act as interfaces for the human users to retrieve and display information.

One particularly well known example of a distributed computing environment is the World Wide Web. In this environment, the Web server computers presently in use typically store data files, or so-called Web pages, in a format known as Hypertext Markup Language (HTML). Web pages are transferred between Web servers and clients using a communication protocol known as Hypertext Transfer Protocol (HTTP). HTML permits the Web servers, or sites, to handle container or document files which reference other files of varying formats. Using HTML, a given Web page may include content information in various formats. An HTML format file may also refer to other files, by including reference information, known as a Uniform Reference Locator (URL), which specifies the location of remote Web servers at which the other files may be located.

Certain Web servers, such as those maintained by on-line service providers such as AMERICA ONLINE ®(AOL®) or Microsoft Network (MSN®), are an increasingly popular way for people to obtain information of interest on the World Wide Web. (AMERICA ONLINE®and AOL® are registered trademarks of America Online, Inc. of Dulles, Virginia. MSN® is a registered trademark of Microsoft Corporation of Redmond, Washington). Indeed, certain Web sites host search engines such as AltaVista®, Yahoo®, and InfoSeek™and thus are exclusively devoted to guiding users through the Web. (AltaVista™ is a trademark of AltaVista, Overture Services, Inc. of Pasadena, California; Yahoo™ is a trademark of Yahoo!Inc. of Sunnyvale, California; and InfoSeek™ is a trademark of InfoSeek Corporation of Sunnyvale, California). These sites are so popular that their operators provide their services free of charge to users of the Web, and support themselves typically by selling advertising space on their Web pages. Thus, an advertiser, for example, a running shoe manufacturer, may contract with a search service such as Yahoo, or an on-line service, such as AOL®, to periodically present its ads on their Web pages in much the same manner that commercials are traditionally purchased from television broadcasters.

Certain tools are presently in use by the providers of such services and advertisers, typically in order to calculate advertising rates. For example, the Web servers at such sites may count the number of times that the Web page containing the advertisement is displayed.

Alternatively, an advertiser may count the number of visits that its own Web page receives as a result of linking

from the original Web page advertisement, i.e., the number of times that users request the URL of the advertiser's Web site via the original Web page on which the advertisement was displayed. In the usual model of user interaction with a Web page, this occurs whenever the user clicks (i.e., selects by a mouse input device) on a hypertext item. In many instances, objects such as graphical images or "GIFs" may be clicked on to activate the hypertext links.

Advertisers, however, would like not only to count a number of "impressions," or how many times their advertisement is seen, but also to find a way to track how effective their ads are in attracting consumers' interest in their products.

Advertisers would also like to find a way to more precisely gauge a user's interest in a product, as well as to entice those users who are casually browsing through the World Wide Web, without actually requiring users to download the advertiser's Web page. In this manner, interest in a particular product or promotion could be gauged directly from data surrounding the initial presentation of the advertisement.

SUMMARY OF THE INVENTION

Briefly, the present invention is a technique for tracking user interactions with the elements that comprise a Web page advertisement. As a result, an advertiser may understand (make inferences as to) what motivates users to pay initial attention to and/or otherwise interact with Web page advertising.

The invention, in particular, tracks any sort of user "micro-interaction" with the advertisement. The user interactions which are tracked, for example, may include mouse movement, mouse clicks, and other mouse activity such as it relates to elements in the ad. These elements may include various display items such as graphics, pictures, or words, or may include user prompting items such as menus, buttons, or slides. Elements also may include defined regions of the advertisement.

The activities monitored may include how long an object is displayed, which objects are selected by a user, which items are considered by a user according to the amount of time the cursor hovers over the items, measuring the time of presentation of an element in various ways, and/or activating hyperlinks.

The tracked interactions are preferably logged to a local memory by a downloadable Web browser applet embedded in the Web page, such as a program written in an interpretive language such as Java™. (Java™ is a trademark of Sun Microsystems Corporation of Sunnyvale, Calif.)

The logged interactions as stored in the local memory file are then sent to a remote server at appropriate times. For example, in the preferred embodiment, the logged interaction information may be included in a "dummy" HTTP GET message sent by the client to the server at the time the applet is taken down, such as when the user requests that a next page be displayed.

The logged interaction information may be flushed in other ways, such as by sending a POST message to the server either periodically or upon occurrence of certain events.

The invention therefore permits the tracking of user interactions with a Web page advertisement before subsequent actions, such as loading the advertiser's home Web page, occur. For example, the applet may intercept multiple interactions such as mouse clicks on objects to further qualify a user before loading a specific one of the advertiser's own home Web pages.

As a result, the advertiser may obtain information about what interests the user without the user having to leave the originally displayed Web page or performing other tasks which are perceived as being cumbersome and/or distracting from what the user was originally doing.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

FIG. 1 is a block diagram of a distributed computing system illustrating a manner of tracking user interaction with a Web page according to the invention.

FIG. 2 is a flow chart of the operations performed by an applet program according to the invention.

FIG. 3 is a typical Web page display illustrating how the invention may prequalify a user.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 is a block diagram of a distributed computing system 10. The distributed computing system 10 includes a number of computers 12, 20 interconnected by a communication media 14. The communication media 14, and in general the distributed computing system 10, may make use of any number of computer networking techniques such as local area networks (LANs), routers, dial-up connections, and/or other data communication techniques to form what has become known as an "intranet" or "internet." In the preferred embodiment, the present invention is employed in what has become known as "the Internet," which is an international computer network linking many millions of computers.

Typically the computers 12, 20 are personal computers, mini-computers, or the like. Certain of the computers in the distributed computing system 10 act as servers 12a, 12b, and are used primarily to store and supply information. One type of server 12a which is in widespread use on the Internet is known as a Web server that provides access to information stored in a form known as a Web page 16.

Other computers in the distributed computing system 10 known as clients 20a, . . . , 20n are typically controlled by one user. The typical client computer 20a includes, as for any computer, a processor 22, a memory 24, and a display 26. The client computers 20 allow a user to view Web pages 16 by "downloading" replica Web page files 40 to the client computer 20a from the server computer 12a over communication media 14. The Web page files 40 enable replication of the Web page 16 on the client computer 20a. The downloading function is specifically performed by a browser program 28, which preferably includes browser program software such as Netscape Navigator™ or Microsoft Internet Explorer™. (Netscape Navigator™ is a trademark of Netscape Communications Corporation of Mountain View, Calif., and Internet Explore™ is a trademark of Microsoft Corporation of Redmond, Wash.) These browser programs include and/or permit the use of embedded interpretive languages 30, such as Java™, that may execute programs that are included in the Web page file 16.

The browser program 28 thus enables the user to create a view of the Web page 16, such as in a window 34 on the display 26. It should be understood that other windows 35 and other programs 36 may relate to other programs 31 that the user is presently running on the processor 22. In order to display the Web page 16, the browser program 28 typically downloads the Web page files to its local memory 24, storing it as a local replica 40. The Web page replica 40 includes various portions such as a Hypertext Markup Language (HTML) as well as other instructions for the Browser program 28 to format the Web page information in the window 34.

The Web page replica 40 also includes Java™ code 44 that includes instructions to be run while the user computer 20a is displaying the Web page.

The display of the Web page replica 40 in the window 34 may include various regions such as a user input area 37 where the user enters addresses of Web pages that he or she desires to view, and menus 38 for other actions associated with operating the browser program 28 itself.

Of interest to the description of the present invention is a Web page replica 40 that contains at least one advertisement area 39. The advertisement 39 on the Web page replica 40 is typically created by the provider of a service or product manufacturer. The advertisement 39 is typically used as an enticement for the user to download other Web pages specifically associated with the originator or author of the advertisement 39.

In accordance with the invention, the Java™ code 44 includes an applet program and data for tracking and logging the activities of the user in memory 24 while the user is viewing the Web page replica 40. The applet program 44 therefore permits the authors of the advertisement 39 to better understand how the users interact with the Web page advertisement in order to provide more effective advertising.

More specifically, the Web page replica 40 includes the Java™ code applet 44 that describes the particular attributes of the advertisement 39. Once the Web page replica 40 begins to display, the applet 44 also begins to execute in order to track and/or log user activities as they relate to various parts or objects of the advertisement 39.

For example, the applet 44 may include information that describes regions 46 of the advertisement 39, a list of visual elements 48 associated with the advertisement 39, and/or user activity definitions 50 that may take place within the context of the advertisement 39. The regions 46 may define areas within the advertisement 39, such as areas devoted to text or graphics. The elements 48 within the advertisement 39 may include various graphical images. The elements 48 may also include user prompts such as buttons, menus, slide bars, radio buttons, and the like.

The list of activity definitions 50 may include various types of user input. The most important user input is typically mouse position, as reflected by the position of a cursor 33, but these may also include other user inputs such as mouse clicks or keyboard inputs.

FIG. 2 is a flow chart of the operations performed by the applet 44 in the process of tracking user interactions with the advertisement 39. An initial state 100 is entered when the applet is first started. This typically occurs when the user requests the display of the Web page 16 and the replica 40 has been downloaded or has at least begun being downloaded from the server 12a.

In the next state 101, the applet program 44 begins execution on the client computer 20a. This is typically in the context of an interpretive language such as the Java™ language executed within the browser 28. However, it

should be understood that the applet may be implemented in other ways, as long as the applet 44 has access to the appropriate user inputs and local memory 24 for the logging of user activities with respect to the advertisement 39.

In the next state 102, the applet 44 establishes a local list of elements and regions on the Web page replica 40 associated with the advertisement 39 that are of concern.

A next state 104 is then entered in which user activities with respect to objects within the advertisement 39 may begin to be tracked by logging information in local memory locations 24 at the client 20. From this state 104, any number of states 106 through 114 and/or state 118 may be next entered for any given element 48.

For example, in state 106, the elapsed time that the element 48 has been displayed on the page is tracked.

In state 108, the fact that the mouse hovered near an element 48, i.e., the fact that the user moved the mouse within a region 46 of the page associated with the element 48 but did not actually click on the element, is tracked.

In state 110, the fact that the user clicked on an element 48 is tracked. It should be noted that this may include the tracking of one or more mouse clicks on one or more elements 48 by making multiple entries in the log 60. Thus, unlike the standard operation of an HTML hyperlink, a single mouse click may not necessarily automatically lead to the loading of the next Web page 16.

In state 112, the applet 44 tracks cursor 33 location at the moment of a mouse click with respect to the element 48.

In another state 114, the applet 44 may adjust the time frame associated with the particular action being logged. For example, when the user initially views a Web page 40, certain activities such as cursor 33 location maybe tracked in short-time intervals such as microseconds. However, other items such as the elapsed time an element 48 is displayed on the page may be tracked in longer time intervals such as seconds. Certain items such as cursor hover time may initially be tracked in a microsecond time frame, and then, depending upon the amount of time the element has been displayed, will switch to tracking a longer time interval, such as seconds.

In state 116, the fact of the user requesting a different Web page is tracked.

It should be understood that, in states 106 through 114, data associated with the various user activities is logged in a portion of the memory 24 associated with maintaining an activity log 60.

Eventually, a state 118 is reached in which the user indicates that he or she wishes to leave the present page 40. This event is typically associated with loading another page in state 116 or may also include the closing of the browser program window 34.

At certain times, state 120 is entered in which the activity log 60 is sent from the local memory 24 by the applet 44 back to a server 12b. The server 12b is typically associated with the advertiser, or an advertisement rating service. This server 12b may or may not be the same server 12a from which the Web page 46 was originally downloaded. The applet 44 may then terminate.

In the preferred implementation of state 120, the activity log 13 is sent to the server 12b via a "dummy" HTTP GET message sent via a "back channel" to the server 12b at the time that the user leaves the present page 40. In particular, this back channel is a second network connection, different from the network connection used to fetch the Web page and download the applet in step 100. The dummy message is encoded as an HTTP GET with interaction log data shared in the GET message in such a way as to appear to be part of

an extended address, for example. Thus, the browser program 28 does not need to perform any special functions or otherwise be modified.

However, the interaction log data may also be sent at other times. For example, while the advertisement is being displayed, the applet may periodically open a back channel connection and send a POST message to the server 12b. Alternatively, certain events may trigger sending the logged interaction data, such as when the user clicks on a particular part of the advertisement.

What is important is that the logged interaction data is eventually flushed to the server 12b, so that the author of the advertisement 39 may occasionally check on the collection 62 of activity logs stored at the server 12b and analyze the data in order to determine the effectiveness of the advertisement 39.

FIG. 3 is a view of a Web page such as that produced by a search engine such as Yahoo®. The advertisement 39 is typically displayed in a defined region on the Web page 40. In this example, the effectiveness of the advertisement 39 associated with the advertiser who is in the running shoe business is being tracked. In the example, the applet 44 tracks how long a user allows the mouse cursor 50 to hover near one of the eight possible displayed selections for running shoes. The applet 44 also tracks the position of the mouse to determine which one or more of the running shoes is selected by user-activated mouse clicks. This information is then sent to the logging server 12b and is used prior to loading the manufacturer's Web page. Thus, the relative interest in a particular type of shoe may be gauged before the advertiser's Web page is loaded or, indeed, a lack of interest, in particular lead or "teaser" items, may be determined prior to the user requesting that the manufacturer's Web page be loaded.

It is now understood how the invention provides various advantages over the prior art. In particular, the invention includes an applet 44 that is downloaded together with a Web page 40 (Web page files) capable of logging the user's interactions with elements 48 on the page. The applet in particular logs user interactions with the page 40 that indicates user interest to an advertiser. By identifying regions on the page and then tracking user activity and relating it to the particular elements or regions on the page, the advertiser may therefore more effectively evaluate the effectiveness of particular objects in the advertising.

By collecting the interaction data locally and then sending them to a server which logs data locally via a back channel, the operation of the client computer or data stored thereon need not be permanently modified.

By tracking multiple interactions before loading the advertiser's own Web page, the advertiser may prequalify a user and hence customize or tailor information to be displayed. In turn, the advertiser may more effectively present the information once the advertiser's Web page is requested by the user.

The user may also be enticed to interact with a Web page advertisement, thereby disclosing information about the user's interests, without their actually requesting the loading of Web pages associated with the advertiser.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

What is claimed is:

1. In a distributed computing system for displaying information, a method comprising the steps of:
 - displaying to a user a representation of an advertisement in electronic form;
 - tracking user interaction with the advertisement;
 - logging the user interactions while the advertisement is being displayed to the user; and
 - sending user interaction log data to a remote server;
 whereby log data is analyzed to determine effectiveness of the advertisement; and
 - whereby advertisement information to be loaded is customized to the user based upon interest of the user determined from the user interaction log data;
 - wherein the step of sending interaction log data additionally comprises the steps of:
 - opening a connection to the remote server; and
 - sending a message to the remote server over the connection, the message containing the user interaction log data;
 - wherein the message is an HTTP GET message; and
 - wherein the user interaction log data is encoded as to appear as part of an extended address field in the HTTP GET message.
2. A method as in claim 1 wherein the interaction log data is sent to the remote server when the display of the advertisement is terminated by the user.
3. A method as in claim 1 wherein the interaction log data is sent to the remote server upon the occurrence of a user interaction with the advertisement.
4. A method as in claim 1 wherein the interaction log data is sent to the remote server periodically.
5. A method as in claim 4 wherein the periodicity of the sending is variable.
6. A method as in claim 4 wherein the periodicity of the sending is logarithmic.
7. A method as in claim 1 wherein the interaction log data includes mouse position and time of mouse hovering over advertisement.
8. A method as in claim 1 additionally comprising the step of:
 - downloading the advertisement from a second server different from the remote server.
9. The method as claimed in claim 1 wherein the step of tracking user interaction with the advertisement is performed periodically.
10. A method as in claim 9 wherein the periodicity of the tracking is variable.
11. A method as in claim 9 wherein the periodicity of the tracking is logarithmic.
12. The method as claimed in claim 1 wherein the remote server is a user interaction log aggregation device.
13. The method as claim in claim 1 whereby the logged user interaction data is stored at a local client computer and sent to a server other than the remote server which logs data locally via a back channel, and the data stored on the client computer is not permanently modified.
14. In a distributed computing system for displaying information, a method comprising the steps of:
 - displaying to a user a representation of a page in electronic form, the page containing at least one advertisement composed of two or more regions, each region containing a visual element of the advertisement;
 - tracking user micro-interactions with each of the elements in the advertisement through the steps of:
 - maintaining a list of elements displayed in the advertisement;

- determining when a screen pointer hover occurs within a particular element of the advertisement, and the screen pointer hover occurring without requiring a user interaction on the element; and
- creating a micro-interaction data record in response thereto;
- logging the micro-interaction data records while the advertisement is being displayed to the user; and
- sending the micro-interaction data records to a remote server;
- whereby the micro-interaction data records are analyzed to determine effectiveness of the advertisement; and
- whereby advertisement information to be loaded is customized to the user based upon interest of the user determined from the user micro-interaction data records ;
- wherein the step of sending micro-interaction data records additionally comprises the steps of:
 - opening a connection to the remote server; and
 - sending a message to the remote server over the connection, the message containing the micro-interaction data records;
 - wherein the message is an HTTP GET message; and
 - wherein the micro-interaction data records are encoded as to appear as part of an extended address field in the HTTP GET message.
- 15. A method as in claim 14 wherein the micro-interaction data records are sent to the remote server when the display of the advertisement is terminated by the user.
- 16. A method as in claim 14 wherein the micro-interaction data records are sent to the remote server upon the occurrence of a user interaction with the advertisement.
- 17. A method as in claim 14 wherein the micro-interaction data records are sent to the remote server periodically.
- 18. A method as in claim 14 wherein the micro-interaction data records include information with respect to screen pointer position within the element.
- 19. A method as in claim 18 wherein the micro-interaction data records include information indicating a time sequence of screen pointer positions within the element.
- 20. A method as in claim 14 additionally comprising the step of:
 - downloading the advertisement from a second server different from the remote server.
- 21. In a distributed system for displaying information content, a method comprising the steps of:
 - displaying to a user a representation of the information content in electronic form;
 - tracking user interaction with the information content;
 - logging the user interactions while the information content is being displayed to the user; and
 - sending user interaction log data to a remote user interaction log aggregation device;
 - whereby log data is analyzed to determine effectiveness of the advertisement; and
 - whereby information content to be loaded is customized to the user based upon interest of the user determined from the user interaction log data;
 - wherein the step of sending interaction log data additionally comprises the steps of:
 - opening a connection to the remote server; and
 - sending a message to the remote server over the connection, the message containing the user interaction log data;
 - wherein the message is an HTTP GET message; and

wherein the user interaction log data is encoded as to appear as part of an extended address field in the HTTP GET message.

22. In a distributed computing system for displaying information, a method comprising the steps of: 5
 displaying to a user a representation of an advertisement in electronic form, the advertisement comprising one or more promotions available from an advertiser;
 tracking user interaction with the advertisement including interaction with the one or more promotions available from the advertiser; 10
 logging the user interactions while the advertisement is being displayed to the user; and
 sending user interaction log data to a remote server; 15
 whereby the user interest in the one or more promotions available from the advertiser is gauged prior to the advertiser's Web page being loaded; and
 whereby the advertiser's information to be loaded is customized to the user based upon the user's gauged interest determined from the user interaction log data; 20
 wherein the step of sending user interaction log data additionally comprises the steps of:
 opening a connection to the remote server; and
 sending a message to the remote server over the connection, the message containing the user interaction log data; 25
 wherein the message is a HTTP GET message; and
 wherein the user interaction log data is encoded as to appear as part of an extended address field in the HTTP GET message. 30

23. A method as in claim 22 wherein prior to the step of sending user interaction log data to a remote server the user interacts with the advertisement without requesting the loading of Web pages associated with the advertiser.

24. A method as in claim 22 wherein the user interaction log data sent to the server comprises information about the user's interest, and wherein the information about the user's interest is obtained without the user requesting the loading of an advertisement associated with the advertiser. 35

25. A method as in claim 22 whereby the log data allows the advertiser to understand how one or more users interact with the advertisement in order for the advertiser to further provide effective advertising. 40

26. A method as in claim 22 whereby the log data is analyzed to determine effectiveness of the advertisement. 45

27. The method as claimed in claim 22 wherein the advertisement comprises regions, visual elements associated with the advertisement, and user activity definitions.

28. The method as claimed in claim 27 wherein the user activity definitions are defined within the advertisement's context. 50

29. The method as claimed in claim 27 wherein the regions define areas within the advertisement including areas within the advertisement devoted to text and areas within the advertisement related to graphics. 55

30. The method as claimed in claim 27 wherein the visual elements within the advertisement comprise one or more graphical images, and one or more user prompts.

31. The method as claimed in claim 27 wherein the list of activity definitions comprises one or more types of user inputs. 60

32. The method as claimed in claim 31 wherein the one or more types of user inputs comprise mouse position reflected by cursor position.

33. A computer program for use in a distributed computing system on a computer readable medium, said computer program comprising:
 program step to display to a user a representation of an advertisement in electronic form;
 program step to track user interaction with the advertisement;
 program step to log the user interactions while the advertisement is being displayed to the user; and
 program step to send user interaction log data to a remote server;
 said computer program further comprising:
 a program step for subsequently analyzing log data to determine effectiveness of the advertisement;
 a program step for customizing advertisement information to be loaded based upon interest of the user determined from the user interaction log data;
 wherein the program step to send interaction log data additionally comprises:
 a program step for opening a connection to the remote server; and
 a program step for sending a message to the remote server over the connection, the message containing the user interaction log data;
 wherein the message is a HTTP GET message; and
 a program step for encoding the user interaction log data to appear as part of an extended address field in the HTTP GET message.

34. A computer program for use in a distributed computing system on a computer readable medium, said computer program comprising:
 program step to display to a user a representation of an advertisement in electronic form, the advertisement comprising one or more promotions available from an advertiser;
 program step to track user interaction with the advertisement including interaction with the one or more promotions available from the advertiser;
 program step to log the user interactions while the advertisement is being displayed to the user; and
 program step to send user interaction log data to a remote server;
 said computer program further comprising:
 a program step whereby the user interest in the one or more promotions available from the advertiser is gauged prior to the advertiser's Web page being loaded; and
 a program step whereby the advertiser's information to be loaded is customized to the user based upon interest of the user determined from the user interaction log data;
 wherein the program step to send user interaction log data additionally comprises:
 a program step for opening a connection to the remote server; and
 a program step for sending a message to the remote server over the connection, the message containing the user interaction log data;
 wherein the message is a HTTP GET message; and
 a program step for encoding the user interaction log data to appear as part of an extended address field in the HTTP GET message.

* * * * *