Computer Networks and ISDN Systems 28 (1996) 1445–1456

COMPUTER NETWORKS and ISDN SYSTEMS

Reducing WWW latency and bandwidth requirements by real-time distillation

Armando Fox *, Eric A. Brewer [1]

University of California, Berkeley, CA 94720-1776, USA

Keywords: Proxy; Compression; Latency; Bandwidth; Mobile

1. Low bandwidth surfing in a high bandwidth world

Today's WWW is beginning to burst at the seams due to lack of bandwidth from servers to clients. Most WWW pages are designed with high-bandwidth users in mind (i.e. 10 Mbit Ethernet) , yet a large percentage of web clients are run over low-speed 28.8 or 14.4 modems, and users are increasingly considering wireless services such as cellular modems running at 4800–9600 baud. A recent study by a popular server of shareware, **Jumbo** [2], revealed that about 1 in 5 users were connecting with graphics turned off, to eliminate the annoying latency of loading web pages [15]. This is of particular concern to corporate advertisers, who are paying for the visibility of their corporate logo.

Since there is no way to optimize a single page for delivery to both high-bandwidth and low-bandwidth clients, some sites, such as [16], offer multiple versions of pages (no graphics, minimal graphics, full graphics). Most sites, however, do not have the human resources or disk space to do this.

Clearly an automated mechanism for closing the bandwidth gap is needed. Several such mechanisms are currently deployed, but all are inadequate:

- Progressive and interlaced GIF and JPEG display a blurry image initially and refine it as more image data arrives. However, for large images, the initial latency is still high, and once the client makes the initial requests, the capability for "choreographing" the download of multiple large images is limited because the data is being pushed by the server, not pulled by the client. The same is true of the **LOWSRC image tag extension** [3] proposed by **Netscape** [4].
- The ALT tag provided in the HTML IMG environment allows a text string to be displayed in place of an inline graphic, but the ALT text usually cannot carry the semantic load of the image it replaces, and most advertisers would readily concede that text is no substitute for corporate logo visibility.
- The Bandwidth Conservation Society [7] has published a number of suggestions for content providers to minimize the size of their images without sacrificing image quality, but their techniques typically provide at most a factor of 1.5–2

* Corresponding author. Email: fox@cs.berkeley.edu.
[1] Email: brewer@cs.berkeley.edu.
[2] http://www.jumbo.com
[3] http://home.netscape.com/assist/net_sites/impact_docs/index.html
[4] http://home.netscape.com

PII S0169-7552(96)00027-X

in compression. This is not sufficient to bridge the order-of-magnitude gap in bandwidth between, e.g., Ethernet and consumer modems.

- A mechanism is proposed in [13] for clients to negotiate for one of several document representations stored at a server. Under the proposed scheme, the server creates a fixed number of representations in advance, possibly with human guidance, and advertises to clients which representations are available. Even if this mechanism were widely deployed (which would require changes to servers), it would not satisfy clients whose connectivity would best be exploited with an intermediate-quality representation not present at the server.
- Caching [9,12,4] and prefetching reduce initial server-client latency and server-cache bandwidth requirements, but do not reduce cache-client bandwidth. We believe that distributed intelligent caching will ultimately be necessary, but not for reducing latency and bandwidth to the client.

The methods described above are ineffective at closing the bandwidth gap because they either require changes at the server (content or control), force additional interaction with the user (e.g. to explicitly select one version of a page), do not allow the user to explicitly manage the available client bandwidth, or require the user to sacrifice graphics altogether. We describe a mechanism that addresses all of these problems: real-time adaptive distillation and refinement.

2. How distillation and refinement can help

2.1. The concept of datatype-specific distillation

Distillation is highly lossy, real-time, datatype-specific compression that preserves most of the semantic content of a document. The concept is best illustrated by example. A large color graphic can be scaled down to one-half or one-quarter length along each dimension, reducing the total area and thereby reducing the size of the representation. Further compression is possible by reducing the color depth or colormap size. The resulting representation, though poorer in color and resolution than the original, is nonetheless still recognizable and therefore useful to the user. Of course there are limits to how severe a degradation of quality is possible before the image becomes unrecognizable, but as we discuss below, we have found that order-of-magnitude size reductions are possible without significantly compromising the usefulness of an image.

Our definition of distillation as *lossy* compression is independent of the specific encoding of the image. For example, GIF is a lossless image encoding format, but the distillation process throws away information in shrinking the image and quantizing its colormap.

As another example, a PostScript text document can be distilled by extracting the text corresponding to document content and analyzing the text corresponding to formatting information in order to glean clues about the document structure. These clues can be used to compose a ''plaintext-plus'' version of the document, in which, for example, chapter headings are in all caps and centered, subsection headings are set off by blank lines, etc. The distilled representation is impoverished with respect to the original document, but contains enough semantic information to be useful to the user. Adobe Systems' *Distiller Pro* package (not to be confused with our use of the term ''distillation'') performs a similar function, constructing a portable document format (PDF) file from a PostScript file.

Clearly, distillation techniques must be datatype-specific, because the specific properties of a document that can be exploited for semantic-preserving compression vary widely among data types. We say ''type'' as opposed to ''subtype'' (in the MIME sense), since, for example, the techniques used for image distillation apply equally well regardless of whether the source image is in GIF or JPEG format.

2.2. Refinement

Although a distilled image can be a useful representation of the original, in some cases the user may want to see the full content of the original. More commonly, the user may want to see the full content of *some part* of the original; for instance, zooming in on a section of a graphic, or rendering a particular page containing PostScript text and figures without having to render the preceding pages.

We use the term *refinement* to refer to the process of fetching some part (possibly all) of a source document at increased quality. We can define a *refinement space* for a given datatype, whose axes correspond to the properties of the datatype exploited by the corresponding distillation technique. For example, some obvious axes for still graphics are scale (as a fraction of the original) and color depth. The source image corresponds to the tuple and $\langle 1, 1, \ldots, 1 \rangle$ in refinement space. Distillation and refinement can then be thought of as parameterized mappings between points in refinement space. The example interface by which a user specifies a desired refinement is application-specific; for example, using a mouse to select a subregion of an image for zooming.

Refinement space for a given datatype may be discrete or continuous. For example, the pixel-dimension refinement axis is (nearly) continuous, but for distilling rich text, we may be able to identify only a relatively small fixed number of intermediate quality representations. For PostScript, these would likely consist of "plain text" (ASCII only with minimal formatting clues), structured rich text (such as PDF or HTML), and original PostScript.

2.3. Trading cycles for bandwidth

Because distillation can be performed in real time, it eliminates the need for servers to maintain multiple intermediate-quality representations of a document: Any desired intermediate representation can be

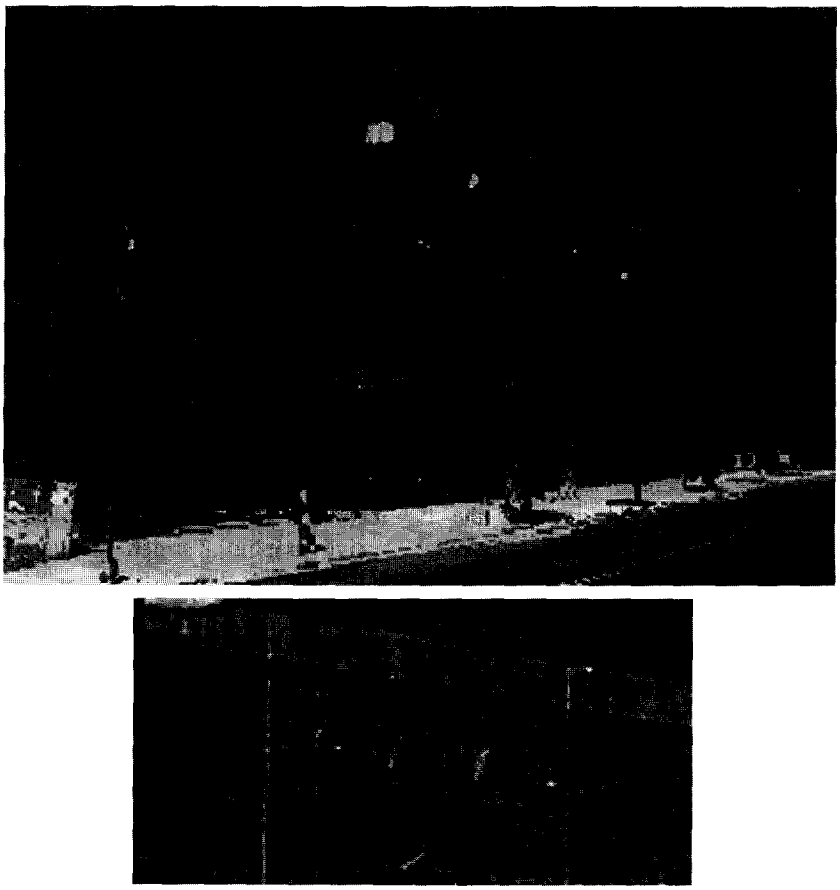

Fig. 1. (top) Soda Hall, distilled image and refinement. (bottom) Distilled to 320 × 200 (17 Kbytes), refinement of writing on building (15 Kbytes).

created on demand using an appropriate distiller. The computing resources necessary for real-time distillation are becoming cheaper and more plentiful, and we have found that at least certain kinds of distillation can be done almost trivially in real time using modest desktop-PC hardware. Distillation and refinement allow us to trade cycles, which are cheap and plentiful, for bandwidth, which is expensive and scarce.

2.4. Using refinement for bandwidth management

As an example of refinement in action, consider a user who has downloaded the image of Soda Hall in Fig. 1 to her laptop computer using a 28.8 modem. The 300×200 image, which occupies 17K bytes and contains 16 colors, was obtained by distilling the original which has pixel dimensions 880×610, contains 249 colors, and occupies 503K bytes. Although the distilled image is clearly recognizable as the building, due to the degradation of quality the writing on the building is unreadable. The user can specify a refinement of the subregion containing the writing, which can then be viewed at full resolution and color depth, as shown in Fig. 1. The refinement requires 15K bytes to represent.

Notice how distillation and refinement have been used to explicitly manage the limited bandwidth available to the user. The distilled image and refinement together occupy only a fraction of the size of the original. The total bandwidth required to transmit them is a fraction of what would have been required to transmit the original, which might have been too large to view on the user's screen anyway. The process of distilling the original to produce the smaller representation took about 6 seconds of wall clock time on a lightly loaded SPARC-20 workstation; the process of extracting the subregion from the original for refinement took less than 1 second.

2.5. Optimizing for a target display

Some notebook computers and PDAs have smaller screens and can display fewer colors or grays than their desktop counterparts, in addition to suffering from limited bandwidth. For such devices, we would like to use the scarce bandwidth for transmitting a distilled representation of higher resolution, rather than using it for transmission of color information in excess of the client's display capability. Intelligent distillation will scale the source image down to reasonable dimensions for the client display, and preserve only the color information that the client can display. Distillation thus allows bandwidth to be managed in a way that exploits the client's strengths and limitations. Table 1 gives a sampling of computing devices with typical display and bandwidth characteristics, with the minimum latencies in minutes and seconds to transfer 5K, 50K and 200K bytes. These numbers serve as zeroth-order approximations for a small inline image, a large inline image, and the total amount of inline image data on a page, respectively.

2.6. Optimizing for rendering on impoverished devices

Some devices, particularly PDAs, have limited onboard computing power and understand only a small number of image formats. It would be painful and slow, for example, for an Apple Newton to receive a GIF image and transcode it to PICT, its native graphics format, for display on the screen. Instead, this transcoding can be done on a more

Table 1
Computing device characteristics

Device	CPU/MHz Typ.	Bandwidth (bits/s)	Display size	Minimum xmit latency, 5K/50K/200K bytes
Apple Newton	ARM 610/20	2400	320×240, 1-bit	0:17/2:50/11:20
Sony MagicLink	Motorola 68340/20	14.4K	480×320, 2-bit gray	0:03/0:30/1:20
Typical notebook PC	Intel or PPC/60-100	28.8K wireline 9600 cellular	640×480 to 800×600, 8-bit color	0:02/0:15/0:60 wireline 0:04/0:42/2:48 cellular
Typical desktop PC	Intel or PPC/60-120	56K ISDN, 10M Ethernet	640×480 to 1024×768, 16-bit color	0:01/0:07/0:29

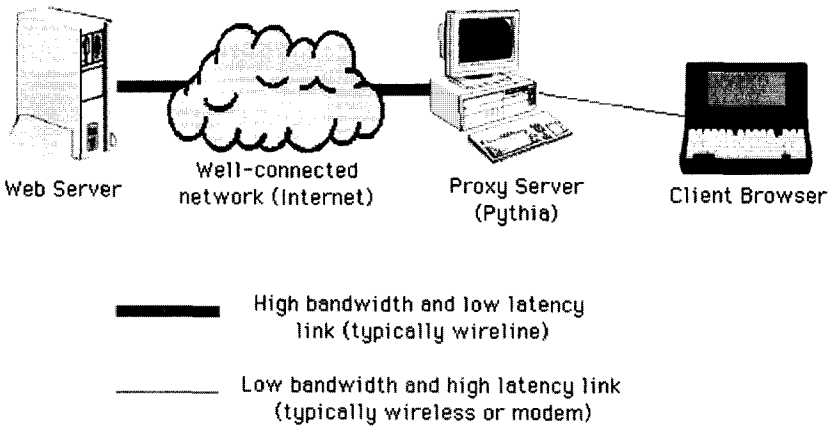

Fig. 2. Architecture of a "proxied" WWW service.

capable desktop workstation as part of the distillation process, *before* the image is sent to the client. The idea of using transcoding to address client limitations has been explored in the Wireless World Wide Web experiment performed at DEC WRL [5].

3. An implemented HTTP proxy based on real-time distillation

We have shown that distillation and refinement provide the user with a powerful mechanism for management of limited bandwidth, without completely sacrificing bandwidth-intensive nontextual (or richtext) content. Since such a mechanism is sorely needed on the WWW, we have implemented an HTTP proxy [10] based on real-time distillation and refinement. We have observed that using our proxy makes Web surfing with a modem much more bearable, and makes Web surfing over metropolitan-area wireless feasible. (Our work on distillation and refinement was originally done in the context of wireless mobile computing.)

Mosaic, Netscape, and other popular WWW browsers allow the user to designate a particular host as a *proxy* for HTTP requests. Rather than fetching a URL directly from the appropriate server, the fetch request is passed on to the proxy. The proxy obtains the document from the server on the client's behalf, and forwards it to the client. The proxy mechanism was originally included to allow users behind a corporate firewall to access the WWW via a proxy that had "one foot on either side" of the firewall. Our proxy, Pythia [5], is intended to run near the logical boundary between well-connectedness and poorly-connectedness.

As a first-order approximation, if we take the majority of the wired Internet to be well-connected and consider a client using PPP or SLIP to be poorly-connected, Pythia can run anywhere inside the wired Internet. The architecture of a "proxied" WWW service is shown schematically in Fig. 2.

The idea of placing a proxy at this boundary has also been explored in the LBX (Low Bandwidth X) project, on which the Berkeley InfoPad's [6] "split X" server is based. The idea of using a proxy to transcode data on the fly was discussed in [8].

3.1. Statistical models for real-time distillation

Pythia works by modeling the running time and achieved compression of distillation algorithms for various data formats. For example, given an input GIF or JPEG and a color quantization factor and scale, the model is used to predict how long the distillation will take and approximately how much compression will be achieved. Our current models provide a ballpark first cut for estimating compres-

[5] In Greek mythology, Pythia was the intermediary who carried a pilgrim's request to the Oracle at Delphi and conveyed the reply back to the pilgrim.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.