TranSquid: Transcoding and Caching Proxy for Heterogenous E-Commerce Environments

Anuj Maheshwari, Aashish Sharma, Krithi Ramamritham
Center for Intelligent Internet Research
Department of Computer Science and Engineering.
Indian Institute of Technology Bombay
Mumbai, India 400076
anuj@iitbombay.org, ash@ee.iitb.ac.in
krithi@cse.iitb.ac.in

Prashant Shenoy
University of Massachusetts
Amherst, MA 01003
shenoy@cs.umass.edu

Abstract

With the advent of the Wireless Internet, the client space has become heterogeneous in terms of device capabilities. To cater to the needs of these devices in E-Commerce applications, smart intermediaries have been developed to increase the user satisfaction by hiding the inherent weakness of some of the small although handy devices like the PDAs and Web-tops. Transcoding has been a popular technique to render data for small devices that have smaller displays, and lesser colour capabilities. But transcoding comes at the cost of caching at the Intermediary.

In this paper, we describe a transcoding and caching proxy that caches objects for heterogeneous client spaces by maintaining separate caches for different categories of clients (PC, PDAs, Mobiles, etc.) and transcoding the lower fidelity versions from the high fidelity variants at the proxies as opposed to fetching the transcoded variants from the server. To achieve this, the proxy keeps the server-directed transcoding information (if provided by the host server) as part of meta data attached to the cached objects and uses this information to convert its fidelity and modes or uses heuristics. Through such an intermediary architecture that serves a heterogeneous client base, we exploit the availability of cached high-fidelity variants of web resources (brought in to serve the requirements of high-end devices like PC's) to serve low end devices, and thereby decrease latency and bandwidth.

1. Introduction

With the everyday computer user adopting the Internet, the opportunity to communicate and interact for business and personal use has exploded to phenomenal levels. As traditional business takes its steps into the electronic environment, the rate of growth of the e-commerce and m-commerce space has been rapid. The development has gone a step further with the advent of pervasive devices, such as cell phones and personal digital assistants (PDA's). All this has led to a greater potential to do business and to share ideas.

As the figures state, the current mobile subscriptions stand at 170 million. They are forecast to exceed one billion by 2003. According to a study published by Strategy Analytics [1] within its strategic advisory service, Wireless Internet Applications, more than 1.5 billion PDAs, handsets and Internet appliances are to be equipped with the wireless capabilities the end of 2004. In money terms, the global mobile commerce market will be worth 200 billion dollars by then and 130 million customers will be generating almost 14 billion transactions per annum. Taking a look at the mobile devices we can expect 70 percent of the new cellular phones and 80 percent of all the new PDAs to have some form of access to the web and further expect 63 percent of the transactions to be generated by mobile devices [2].

All the above is based on the hypothesis that mobile phones and small handheld devices would be able to provide the quality of service and features desired by a common user to carry out day-to-day business activities. An enabling and essential feature for the same is the ease to access the Internet at a reasonable user satisfaction [1]level. But then one needs to understand that there are several challenges in making mobility a widespread reality. Many problems like lower bandwidth, higher error rates, frequent disconnections, smaller displays are common across mobile phones and PDAs. Another important issue here is that of *heterogeneous client space*.

[1]User satisfaction is the perceived experience of a typical user during Internet surfing.

Heterogeneous client space exists when data is available to users across a growing number of destinations - laptops, desktops, kiosks, automobile browsers, cellular phones, pagers, pocket PCs, Palm OS devices and other handheld devices. Current trends suggest that the evolution of the Internet is towards more and more heterogeneity. In heterogeneous client environments, each destination brings its own unique requirements. Devices present different graphical, bandwidth and display requirements, and may support a variety of data formats (e.g. XML, WML, cHTML, SVG). They have different processing capabilities and constantly evolving standards. Such a heterogeneous client environment is very distinct from the prevailing homogeneous client environments in which the only client type is a personal computer. Every response in a heterogeneous environment is not only a function of the URL but also depends significantly on the kind of the device making the request. This additional constraint in such environments forces a rethink of certain prevalent technologies, which provided performance benefits in homogeneous environments completely useless. One such technology is the caching of web objects.

There is no doubt that caching has increased user satisfaction tremendously by serving web objects locally from a proxy that is near the client. We believe, caching can also provide similar benefits in heterogeneous client environments though this requires additional features at the intermediary and standard protocols for the end user to communicate. One of the fundamental problems that appear in enabling caching for heterogeneous environments is storing of multiple variants of the same object generated because of the transcoding operation at the server side or at an intermediary before the caching server. Presently, all transcoding engines mark transcoded content as un-cacheable. We believe this is unnecessary and leads to wastage of bandwidth and increases latency in a response that needs transcoded data since the content now travels across the Internet when it could very well be served from a location near the client.

We present a novel caching and transcoding system called *TransSquid* that is an extensible and intelligent intermediary for the heterogeneous client environments. *TransSquid* is a modular framework that enables caching in heterogeneous environments by maintaining a multi-level cache and linking it with the transcoder.

This paper is structured as follows - we first present our technique and then discuss its implementation in detail. Then, we look at related techniques that help in enhancing the user satisfaction through mobile devices like PDA's and mobile phones and compare them with our work. We end with a discussion on the open issues and also present the scope of *TransSquid.*

2. Caching in Heterogeneous Client Environments

To place things in perspective, we first discuss caching technology, as it exists today. Cache technology provides for the three primary characteristics - scalability, availability and responsiveness. The fundamental principle of a cache is to provide frequently requested content locally and reduce the latency for the request to be serviced.

With the advent and the widespread usage of the Internet, caches have been used to store web objects. They could be deployed at the end point or as an intermediary between the client and the server. The essential idea is still the same, to provide faster access to content. On the whole, all caching technologies, whether at the end point or at an intermediate location revolve around reducing latency by avoiding slow links between client and origin server. They try to make up for slow connections and network congestion. For ISPs and Content Providers, caching further reduces the overall traffic on the networks and allows them to offer high-performance distribution at low cost.

For a typical web request, caching functionality is embedded into the HTTP protocol. A HTTP [9] transaction begins when the client sends a request with the URL, using a series of HTTP *header requests*. This request is triggered by an intermediary caching proxy that checks for the presence of the requested object in the local cache. If the object is present in the cache, the proxy would append an *if-modified-since* request header with the request. The server responds to the typical request by first sending HTTP *response headers* that contains information about the requested content eg. *date*, *content_type*, *last_modified*, and *content_length*. In case a *if-modified-since* request header is present in the request, the server simply sends a "`304 Not Modified`" reply if the object at the cache is the most recent version else, the client has to download the web object from the end server. When the client request is furnished from the cache, it is termed as a HIT. If the cache does not have the latest version of the requested object and the object is got from the end server, it is termed as a MISS. All caching algorithms try to maximize the probability of a HIT given the limited storage space available with the cache.

As discussed earlier, the constraints in a heterogeneous environment are very different from that of the homogeneous environment. In the next couple of paragraphs, we shall discuss in more detail, the issues and imperatives in such environments from the angle of caching.

Transcoding and content negotiation lead to multiple variants of a resource or a web object that differ in modality and fidelity [2]. For example, a typical PDA client would be

[2] A media resource can be translated to different modalities, such as text to audio, or video to images. Where as different versions within the same modality, occurring for example due to, image compressions, text summarizations and video abstractions are the fidelity variants of a resource.

satisfied with a lesser fidelity variant than the variant for the same resource for a Workstation client that would demand rich features.

Table 1. Client Capabilities

Client Type	Bandwidth	Display Size	Color Device	Storage
Work-Station	10 Mbps	1280x1024	24-bit color	20 Gb
Color PC	56 Kbps	1024x768	24-bit color	10 Gb
PDA (high-end)	14.4 Kbps	320x200	8-bit color	16 Mb
PDA (low-end)	9.6 Kbps	320x200	Greyscale	8 Mb
WAP-enabled Mobile Phone	20 bps	132x176	b/w	2 Mb

Essential requirements for intermediaries, especially caching proxies in such environments are, recognition and persistence of the different variants of the same resource. For the former, standards can help in the identification and resolution of variants through attached headers. It is the persistence of multiple variants of the same resource that would require intelligent caches that have a mechanism to categorize them according to one or a set of parameters (namely fidelity, requesting client, content-type) and store them accordingly in a way that makes retrieval fast and simple. This makes the design of such a cache both interesting and challenging.

For the identification of the client type in an heterogeneous environment, it is imperative for the intermediary proxy and the end server to understand the Client Capabilities and Preference Profiles (CC/PP) [4]. CC/PP is a collection of the capabilities and preferences associated with user and the agents used by the user to access the WWW. For an intermediary caching proxy, understanding CC/PP would help it in the classification of clients so that variants of the same resource could be appropriately made available to them. The service and the content provided by a caching proxy would depend entirely on how well the proxy can understand the requirements of a client and thereby, on it ability to deliver the closest possible variant available. CC/PP, thus gives a standard for such communication.

Intelligent caching proxies in a heterogeneous client environment could have the functionality to exploit the heterogeneity through the conversion of high fidelity variants of a resource to low fidelity variant to serve clients at the lower capacities. A typical example of this would be the conversion of high fidelity JPEG image to a low fidelity GIF image in reduced colour and half the dimensions to serve a low resolution PDA client. This would imply that a higher fidelity variant of a resource could be used to satisfy the needs of clients that have lower capabilities, through local transcoding of content at the caching proxy itself.

For this, a transcoding module that is tightly coupled with the caching engine is needed at the intermediary. The above discussion highlights the requirements for a caching proxy in heterogeneous client environments. To summarize, a caching proxy can be designed to:

1. Recognize multiple variants of a resource and categorize them in the cache accordingly.

2. Understand the Clients Capabilities and Preference Profiles (CC/PP) to facilitate client recognition.

3. Manipulate fidelity or modality (or transcode) of variants whenever possible to provide better service.

3. TransSquid: A Caching and Transcoding Intermediary

Our novel technique is a multi-level caching solution for the emerging heterogeneous client environments. The *TransSquid* architecture is designed as a smart intermediary that can cache and transcode web objects in an environment where the client requests have to be serviced intelligently according to the client capabilities. *TransSquid* tries to solve the problem of caching in a heterogeneous client environment by taking a *client centric approach* for categorizing different variants and then storing them in a multi-level cache on the basis of fidelity and the modality.

As can be seen in Table 1, there are an ever-increasing myriad of devices for accessing the web. Given this, it would not be sensible to provide for caches for each type of device as the current base of these devices is high and is increasing because of innovation and newer players. If one cache were provided for each device and if a cached object that serves a Windows CE based iPAQ, it would not be able to support a Palm device, which could share the same data given their capabilities. Separate caches for each device would lead to low and perhaps ineffective HIT rates in the cache, which defeats the purpose of the cache itself. *TransSquid*, therefore provides a limited level caching architecture, by dividing the client space into a limited number of (say three) categories based on the capabilities. A client device is a member of one and only one of the 3 categories depending on its capabilities like display size, colors, storage and bandwidth of connection. All web objects ac-

cessed from a client device are stored in the cache that the client has membership to.

The 3 major categories that we divide the client space are:

1. **High Capability Clients:**

   The clients that fall in this category are Personal Computers, Work Stations and Laptop Computers. These devices have large storage capacities (typically more than 64 MB RAM and 10 GB or more disk space), large screen size (640 x 480 pixels or above), multimedia support, and good processing power. These devices have functionality to view video, audio and high resolution images. Content available on the Internet is default for such kind of computing devices.

2. **Medium Capability Clients:**

   Portable Computers like PDA's and WebTops fall in the category of Limited Capability clients. Typical examples of such devices are the Palm Pilots and iPAQ's. The demand for such portable and handy devices has grown exponentially. These devices have smaller screen size (typically 320 x 200), limited colors, lesser processing power (100 Mhz) and are connected to the Internet through slow wireless links. Some devices in this category also have audio functionality.

3. **Limited Capability Clients:**

   These are very low-end devices that have been used to surf the internet only for score updates, news headlines and other text-only or have very low graphics support. Some models of mobile phones have come up with larger screens, but inherently the data transfer to these devices is very slow. SMS - a popular technique for data transfer to mobile phones has a bandwidth of 20 bytes per second. Though with new Wireless Transfer Protocol (WAP) and other efforts from the research community this has improved, surfing using a the mobile phone remains slow.

As can be seen, we take a middle path. We justify our categorization into a limited sets by saying that broadly the requirements of all clients in a particular category are the same. Though, certain differences in capacities might exist, we take a broader view for user satisfaction. The overall picture is represented in Figure 1. Any necessary modification could be done on the fly. Our experiments suggest that PDA devices with Windows CE and the Palm OS (the two major players in the small device OS market) give almost the same visibility and feel for content. Though different transcoding engines have proprietary techniques for transcoding, we feel that the difference in the rendering for these clients that are close to each other in CC/PP is not very

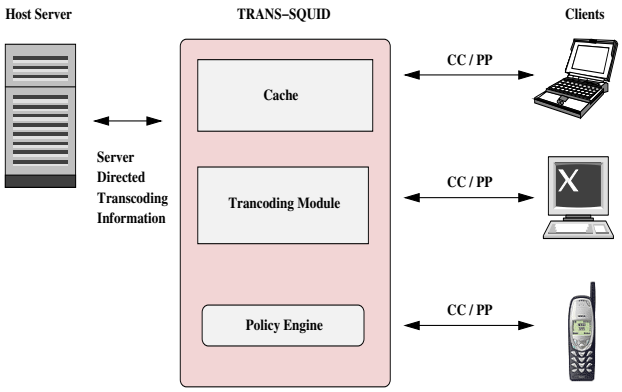

Figure 1. Basic scenario for TransSquid

high, and hence the same cached object could provide the necessary level of user satisfaction.

If certain transcoding needs to be done on an object in a cache to suit the requirements of a request made from the client in the same category - it could be done on the fly or at the client side, but the performance benefits provided by giving a Cache HIT would outweigh the less optimal solution in transcoding.

Intra-cache communication between the different levels in *TransSquid*, allow the low end users to benefit from the availability of high fidelity content. Our assumption is that normally a high fidelity variant can be converted into a lower fidelity variant and hence objects of a higher fidelity cache can respond to a request for a lower fidelity cache by local transcoding. This is broadly true for images and HTML data whose fidelity can be changed by decreasing resolution or by removing unnecessary information. We term this as Partial HIT and discuss this concept in our proceeding discussion.

3.1 Notion of Partial HIT

HIT and MISS are the two primary events that take place when the client sends a request for a object to the network. Depending on attributes like *if-modified-since* and client preferences, the object is either returned from the cache or fetched from the server, as seen in our earlier discussion on caches.

In the *TransSquid* architecture, when a client with low fidelity requirements, makes a request, for which the cache already contains a higher fidelity variant but no object in the cache that the client maps to exactly, the cache returns a Partial HIT. In such a case, the cache sends the object to the transcoding module of the *TransSquid* architecture, which uses the information available in the meta-data of the Object, the *content_type* and Characteristics of the Object as its input to return a suitable variant of te resource. If the meta-data contains the directive given by the host server (based

on information semantics) then this information is used for transcoding. This is called *Server Directed Transcoding* [21].

A Partial HIT is more time consuming than the HIT in which the object is uploaded from the disk and served to the client. The additional time is primarily consumed in - firstly, trying to determine the fidelity and the modality of the variant from the variant available in the cache, and secondly, time taken to perform the actual transcoding operation. This is still an order above the MISS case, in which the object is fetched from the server, and possibly transcoded at an intermediary as well. Figure 2 shows the relative times taken by MISS, HIT and a Partial HIT.

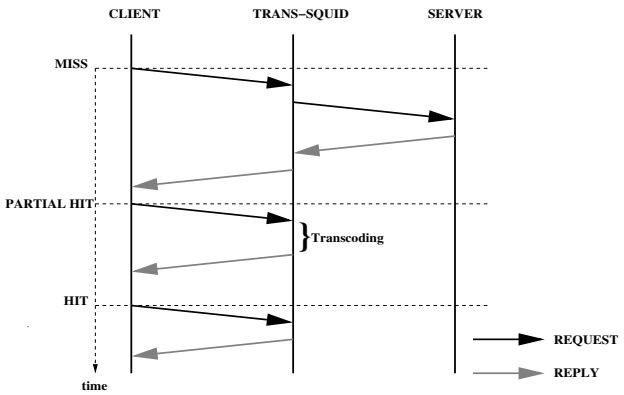

Figure 2. Concept of a Partial HIT

In this section, we saw the *TransSquid* architecture and the problem that it tries to solve. Our solution is based on certain assumptions that are based on our understanding of the WWW and also the heuristics and rules of thumb available for the Internet in general. In the next section, we discuss the implementation of *TransSquid*.

4 Implementation of TransSquid

4.1 Architecture

The *TransSquid* is designed as a modular framework where each module has a specific function. The major parts of the *TransSquid* are:

1. Multi-level Caching Module
2. Transcoding Module
3. Client Side Module
4. Policy Engine

A schematic representation of the architecture is shown in the Figure 3. The Multi-level Caching Module is structured in the form of three caches that store the web objects

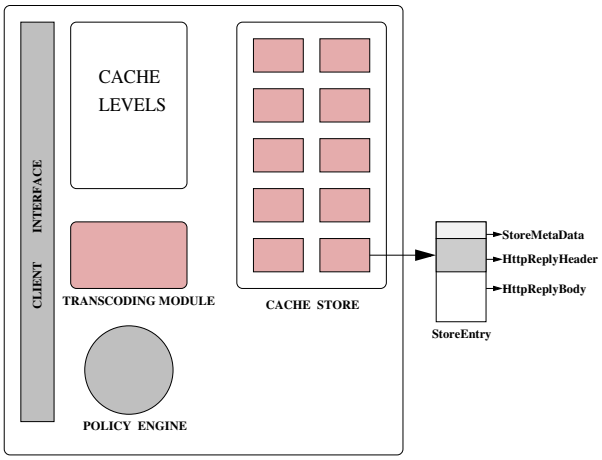

Figure 3. Architectural overview of *TransSquid.*

according to the requesting client type. Each level functions as a separate cache though they have a central persistence. The cache replacement policy for the cache is FIFO. The cache replacement policy by itself in such architecture is an interesting and unexplored issue for research.

These caches are hierarchical as the caches serving the high capability clients can also serve requests from devices like mobile phones and PDA's after passing the objects through the transcoding module. As discussed later, the transcoding module renders the objects for lower capability devices.

The Transcoding Module in our implementation is a heuristic based web object processor, which recognizes two types of objects - images and text. For images the transcoding modules applies functions like reducing the dimensions, decreasing the number of colors and decreasing the quality factor of JPEG images. The function to be performed on an image is chosen through the policy engine or by using any server directed information appended with the web object. For text, we apply simple techniques like removing unwanted parts like advertisements, buttons and unnecessary information replacing them by simple text links.

The Policy Engine provides the Transcoding Module with information about current modality and fidelity of a web object and also suggests the function that should be carried out to render it for other devices types. Again, this is based on heuristics like *content_type*, size, and dimension, in the case of images.

The Client Side Module is the interface between the client and the proxy. It is used for intercepting the client requests and mapping it to the correct cache. The client side maintains a client specific state so that a client need not advertise its capabilities and preferences (CC/PP) every

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.