Differentiated Multimedia Web Services Using Quality Aware Transcoding

Surendar Chandra, Carla Schlatter Ellis and Amin Vahdat
Department of Computer Science, Duke University, Durham, NC 27708
{surendar,carla,vahdat}@cs.duke.edu

Abstract—**The ability of a web service to provide low-latency access to its contents is constrained by available network bandwidth. It is important for the service to manage available bandwidth wisely. While providing differentiated quality of service (QoS) is typically enforced through network mechanisms, in this paper we introduce a robust mechanism for managing network resources at the application level. We use transcoding to allow web servers to customize the size of objects constituting a web page, and hence the bandwidth consumed by that page, by dynamically varying the size of multimedia objects on a per-client basis. We leverage earlier work on characterizing quality versus size tradeoffs in transcoding JPEG images to dynamically determine the quality and size of the object to transmit. We evaluate the performance benefits of incorporating this information in a series of bandwidth management policies. We develop metrics to measure the performance of our system. We use realistic workloads and access scenarios to drive our system.**

The principal contribution of this work is the demonstration that it is possible to use informed transcoding techniques to provide differentiated service and to dynamically allocate available bandwidth among different client classes, while delivering a high degree of information content (quality factor) for all clients.

Keywords— **Quality Aware Transcoding, Differentiated Web Service, Quality of Service**

I. Introduction

THE web is emerging as the primary data dissemination mechanism for a variety of applications. A primary goal of a web service is to provide low latency access to its contents. However, during times of high demand, this goal is compromised by locally available network bandwidth.

Caching at web proxies is one traditional technique to address bandwidth limitations by replicating objects. However, much web content is dynamically generated (maps, stock charts, etc.) or un-cacheable (sites selling access to images or movies). With the availability of low cost commodity compute servers, the network costs far outweigh the cost of other computing resources designed to consume such bandwidth. Web services do not want to add expensive bandwidth without clear policies to allocate the bandwidth. Hence, we focus our attention on differentiated allocation of network bandwidth.

The following important trends illustrate the need for a differentiated QoS system:

E-commerce: E-commerce and subscription based services such as Proflowers and ESPN want to prioritize their consumers based on their subscription status, prior access history and their current status (e.g., a customer with a purchasing history or one with a full shopping cart). To retain their paying customers, these sites need to maintain better QoS for the preferred customers. In order to convince new users to subscribe, these sites need to provide quality teaser objects.

Web hosting: As the complexity of maintaining a web site increases, there is a trend towards hosting services (e.g., [1]) that maintain web pages on behalf of their customers. These hosting services charge their customers based on the size of the web site and the aggregate consumed bandwidth. One of the problems encountered by web hosting services is that "flash users" in another web site hosted by the same hosting service can degrade the performance for all the other locally hosted sites. Current web hosting services use a laissez-faire approach in managing their bandwidth. However, customers of these hosting services will demand performance guarantees, effectively forcing the servers to provide differentiated services.

In such a scenario, differentiated service can allow the system to provide better service for certain customers, based on their status and the current network environment. Differentiated service enables:

- Web services to dynamically allocate the available bandwidth among different user classes,
- Subscription services to provide different versions of contents to clients based on customer status (subscriber versus non-subscriber),
- Web hosting services to share their bandwidth for different classes of hosted clients,
- E-commerce sites to allocate their bandwidth to customers who are making a sale, and
- Flexibility to redirect unused preferred resources to non-preferred customers.

By some estimates [2], about 77% of the data bytes accessed across the web are from multimedia objects such as images, audio and video clips. Of these, 67% of the data are transferred for images. Hence, we focus our attention on image data.

Our approach to providing differentiated QoS is to transmit variations of the same multimedia object at different sizes, allowing some control over the amount of bandwidth consumed in transmitting a page to a particular client. The key insight behind our work is that allocation of critical network resources can be done dynamically at the application level in response to client access patterns.

The feasibility of our differentiated service scheme depends on the availability of a range of variations for the content so that the server can choose the correct variation for the current network operating environment. While the content provider can manually provide a number of different variations for use by the system, an automatic technique allows the system to dynamically adapt to variability in network performance and client characteristics.

We use transcoding to dynamically create variations of the same multimedia object. Transcoding is a transformation that is used to convert a multimedia object from one form to another

(frequently trading off object fidelity for size). By their very nature, multimedia objects are amenable to soft access through a quality-versus-size tradeoff.

For transcoding to provide the degree of control needed to deliver differentiated service, we need to understand its inherent tradeoff characteristics: the information quality loss, the computational overhead required in computing the transcoding and the potential benefits of reduced bandwidth requirements. To illustrate for one specific case, our earlier work [3] characterized the information quality tradeoffs, the computational requirements and the potential size reductions for transcodings that change the JPEG [4] compression metric. In our companion work [5], we analyzed the transcoding characteristics of web images and showed that changing the compression metric of a JPEG image is a promising transcoding for JPEG images accessed on the web.

In this paper, we shift our attention to server performance. We explore the potential benefits and overhead in providing differentiated service for different classes of users and better utilization of the scarce network resources available to the server by performing informed transcoding. This implies several sub-problems: the first problem is to precisely define the performance metrics that can measure the performance of our system. Next, we need to identify realistic access scenarios and workloads so that the results are valid for a range of scenarios. Our system will be successful if it can perform adequately for our metrics under realistic scenarios.

Towards this end, we develop metrics to measure our system performance. We use realistic workloads gleaned from popular web sites to drive a modified Apache web server [6]. While we use adjusting the JPEG compression metric as the informed transcoding technique in this paper, we believe that our results are equally valid for any transcoding with well understood tradeoff characteristics.

The principal contribution of this work is the demonstration that an application-level technique, informed transcoding, can provide efficient and dynamic differentiated service. We show how a web server can dynamically allocate available bandwidth among different client classes, while delivering a high degree of information content (Quality factor) for all clients. Our results make it possible for a heavily loaded web server to selectively reduce the information Quality factor of its multimedia images without resorting to ad-hoc service denials.

The rest of the paper is organized as follows: Section II reviews our previous work as the necessary background. Section III outlines the experiment objectives and design constraints, the system architecture, the workload used and implementation details of our system. The experimental results are described in Section IV. Section V places our work in context to other related work with conclusions and future work in Section VI.

II. Background: Quality Aware Transcoding

Quality aware transcoding is the enabling technology for our research. Transcoding operations are often performed to fit an object to the characteristics of the display device. Images have been transcoded to thumb-nails, gray-scale, progressive formats as well as transcoded to textual information. For example, full color JPEG images are transcoded to a bitmap form to reduce latency for modem users [7]. Our focus is on transcoding to reduce bandwidth requirements on the server. Very little work has been done in determining the level of transcoding needed to be effective at bandwidth reduction and in quantifying the actual information loss and computational characteristics of those transcoding operations.

In our companion work [5], we analyzed the transcoding characteristics of web images and showed that changing the compression metric of a JPEG image is a promising transcoding for JPEG images accessed on the web. Hence, we characterized the tradeoffs inherent to a transcoding that changed a JPEG compression metric, such as the JPEG Quality factor [3]. Reconstructing the original Quality factor that used to produce the image is necessary so loss in quality becomes meaningful. Using the quantization tables stored in the JFIF [8] headers, we developed an algorithm to predict the Independent JPEG Group's (IJG) [9] equivalent of the JPEG Quality factor for images compressed using popular JPEG compressors from IJG, Adobe Photoshop and Paint Shop Pro. We utilized results showing that the information quality loss directly corresponds to the change in the JPEG Quality factor [10], [11].

Next, we characterized the computational overhead and the expected change in image size for a particular transcoding. We showed that the computational requirements for a transcoding that changes the JPEG Quality factor do not depend on the actual Quality factor change, but on the sum of Minimum Code Unit (MCU) block counts for all the different color space components. We showed that this transcoding can be performed entirely in the frequency domain, avoiding computationally expensive Fourier (FFT) transformations.

We also developed a heuristic to predict if an image will transcode efficiently, wherein it loses more in size than in image quality. We empirically showed that images with high coefficients for low frequency components as well as images with initial JPEG Quality factor greater than 80 can transcode images efficiently at a significantly better percentage than the base case of all the images.

These previous results are the enabling technology for our research effort.

III. Experiment Objectives and Design

A. Objectives

Our experiments are designed to answer the following questions:

- For a web service offering differentiated service, can quality aware transcoding allow the web service to better manage its available bandwidth?
- For a web server offering differentiated service, can quality aware transcoding allow the web service to provide differential service for preferred and ordinary clients?

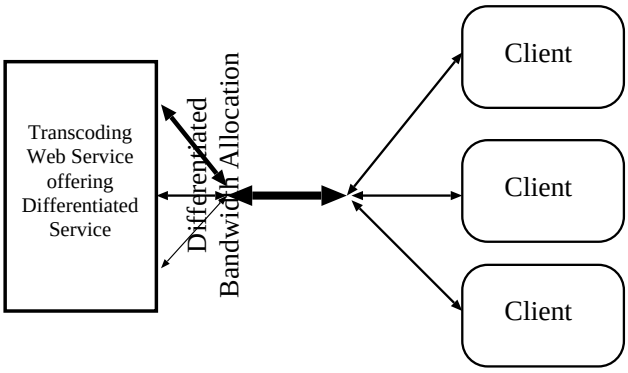


Fig. 1. System Architecture

B. System Architecture

The system that we envision is described in Fig. 1. The web server uses transcoding to provide differentiated service. The transcoding required to provide differentiated service for static images is performed once and the results of the transcoding are cached for reuse. Even though the transcoding adds a computational and storage overhead, as quantified by [3], in this study we are mostly concerned with the limited, expensive network bandwidth to the Web server.

C. Performance Measures

The goal of a web server that offers differentiated service is to serve as many users as possible at sufficiently attractive levels of quality and latency to gain and retain their business. The chief constraint to the ability to serve many users is the limited net bandwidth available to the wide area network. Web servers are typically served by multiple T1, T3 or LAN links.

In such an operating environment, the following measures capture the performance of a web server.

- *Bandwidth Consumed*: Bandwidth control is the primary constraint for our research. The goal is to maintain the bandwidth consumed within pre-defined levels, regardless of the Web traffic.
- *Image Quality Factor*: Since transcoding trades off image information quality for size, the Quality factor of images served gives an indication of the quality tradeoff. The goal is to maintain as much information quality as possible, within the constraints of the available network bandwidth.
- *Client Access latency*: Client experience not only depends on the information quality factor, but also on the latency of accessing the images. Using results from human factors research, we use ten seconds as the threshold on acceptable client latency. Lower latency is important for e-commerce sites that want to attract customers.

D. Web Service Policies

For our experiments, we explore the following policy alternatives for bandwidth control.

- *Traditional:* First, we analyze the performance of the base Web service without any bandwidth control mechanisms. For our experiments, we use the unmodified Apache Web server for the base case.
- *Modbandwidth:* Next, we analyze the performance of a bandwidth limiting web service that prioritizes the network packets, delaying packets associated with low priority requests. For our experiments, we use the Apache mod_bandwidth module to restrict the bandwidth consumption.
- *Denial:* Next, we analyze the performance of a bandwidth management scheme that temporarily denies requests under heavily loaded conditions. The users are expected to defer attempting to access these denied resources until the server becomes less loaded. For our experiments, we modify the Apache Web server to return the HTTP [12] error code "503: Service Unavailable" to deny requests during heaving loaded conditions.
- *Transcoding:* Finally, we analyze the performance of a web service that offers differentiated web service by transcoding an image to a number of variations. The choice of the number of variations is a compromise between fine control of object sizes and storage requirements to store the variants. A goal in image transcoding is to ensure that that any loss in image quality is *efficient*, defined as a transcoding that loses at least as much in image size as the loss in image information quality [3].

E. Experimental Workload

The effectiveness of our study depends on the realism of the workload presented to the system under test which includes the requests as well as the JPEG images that are used as targets of the requests. The requests should capture the behavior of a typical, heavily loaded web service. The kind of web server that we envision has access to high quality images that can be automatically transcoded by the server.

In order to generate such realistic requests, we need to first develop the access trace to the web service. Then, we need to develop the JPEG images that are the targets of these requests. Next, we need to develop ways of generating requests from preferred customers, so that the system can provide differentiated service.

E.1 Access Traces

The realism of our experiments depends on the accuracy with which we can model typical client accesses to a popular web server. We need a client access trace that captures the client request arrival times so that we can compare our system to existing approaches. Unfortunately, popular web sites consider this access information proprietary and hence do not want to share this information.

Hence, we develop a synthetic access trace to closely approximate observable access patterns. We sample the accesses to popular web servers by analyzing accesses made via the NLANR [13] proxy caches. For our experiments, we ana-

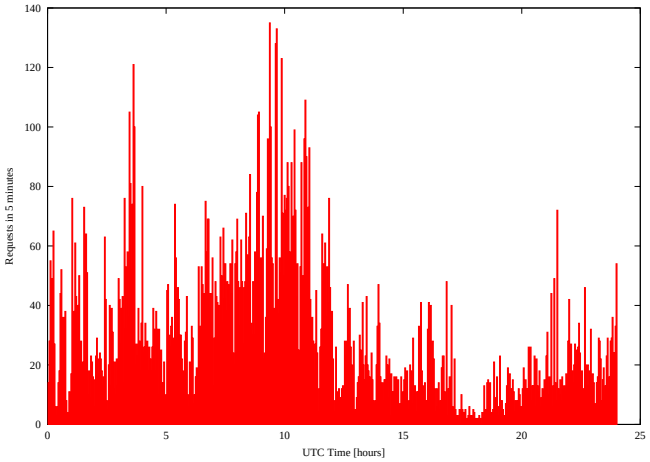


Fig. 2. Accesses to geocities.com via IRCache @ Boulder, CO (Mar 23, 1999)

lyzed the proxy traces collected on March 23, 1999 from the NLANR proxy at NCAR and NCSA. NCAR predominantly serves the .com US domain and NCSA serves .net, .edu and .org US domains. For our experiments we use the accesses to geocities.com, which was ranked among the top ten popular Web sites by Nielsen Netrankings [14].

The number of accesses to geocities.com, via the NLANR proxy, measured in 5 minute intervals, is plotted in Fig. 2. We measured an average traffic of 0.2 accesses/sec. From Fig. 2, we note that the accesses show wide variability within short intervals of time. We noted similar spikes in all the other popular web sites accessed through the NLANR proxy.

However, over a 24 hour period, the accesses show an overall trend of higher accesses during evening hours and lower accesses during early morning hours. Earlier work [15] identifies similar hourly variations.

We use these sampled accesses to model a synthetic access trace for our work. Based on the characteristics in Fig. 2, our trace changes its access rate every 30 seconds to model the diurnal influence on access rates. For the experiments described in Section IV, we scale our access trace to approximately 12,500 seconds (3.5 hours).

E.2 Image Collection

The next parameter of our workload is the composition of the collection of actual JPEG images that are served for requests to the web service.

In keeping with the e-commerce thrust of our work, we used 3531 JPEG images totaling 38 MB, downloaded from BMW.com, Proflowers.com and Starwars.com. Each of the accesses in our synthetic trace (described in the Section III-E.1) is assumed to to a random element within this image set. We are currently investigating more realistic models for web page contents and structure. However, the results presented in this paper validate a web server's ability to control consumed bandwidth and to provide differentiated quality of service through transcoding.

For the JPEG images in our collection, we plot the cumulative distribution of the image size and the original JPEG Quality factor in Fig. 3. From Fig. 3(a), we note that the images are of high quality, 60% of the images have JPEG Quality factor higher than 70. From Fig. 3(b), we note that 40% of the images are bigger than 10 KB.

In our system, the images are transcoded by the server to a number of images of different Quality factor values. Next we perform experiments to measure the number of variations possible for the images in our workload.

For the images in our workload, we transcode the images to a percentage of the original Quality factor and measure the percentage of images that transcode efficiently. Recall that an efficient transcoding loses no more in quality factor than in size (as a percentage of the original value of each parameter) [3]. The results are plotted in Fig. 3(c).

From Fig. 3(c), we note that there is a significant drop in the percentage of images that transcode efficiently when the images are transcoded to a Quality factor of 30% or less of the original image Quality factor. Hence, for our experiments, we transcode the original images to Quality factor values of 90%, 80%, 70%, 60%, 50%, 40% and 30% of the original image Quality factor. For our workload, the transcode cache takes 165 MBs, a nominal amount of storage by today's standards.

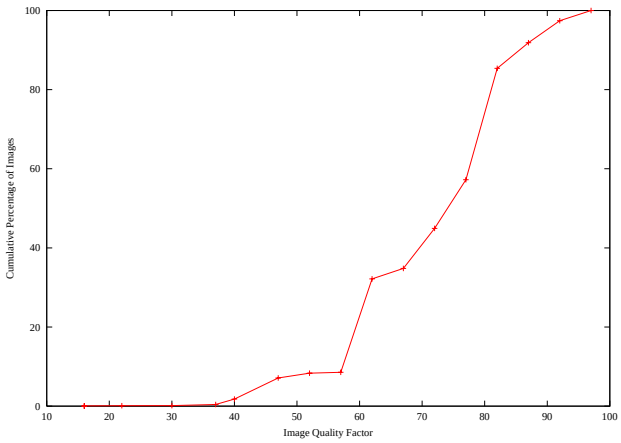
E.3 Client Classes

The next workload parameter we consider involves developing ways to generate accesses from different client classes. For our experiments, we modify http_load to generate a configurable percentage of requests from different client classes. The http_load informs the server about the class that a request belongs to using custom HTTP headers.

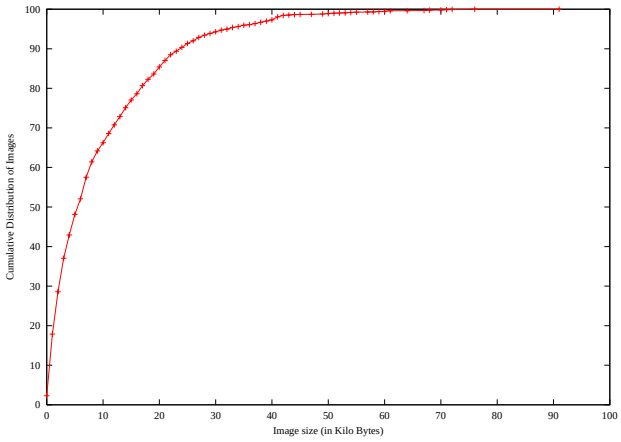
F. Implementation Details

For our experiments, we used a 450 MHz Pentium-III on an Asus P2B motherboard with an Intel 440BX chipset, with 512 MB of main memory, running FreeBSD 4.0. We reconfigured the FreeBSD kernel with more *mbuf*s to handle the higher network loads. The image collections and the server's internal transcoded images were downloaded from the original web servers and stored on two separate, dedicated 21 GB IBM DeskStar 22GXP drives on separate Promise Ultra/33 IDE channels.

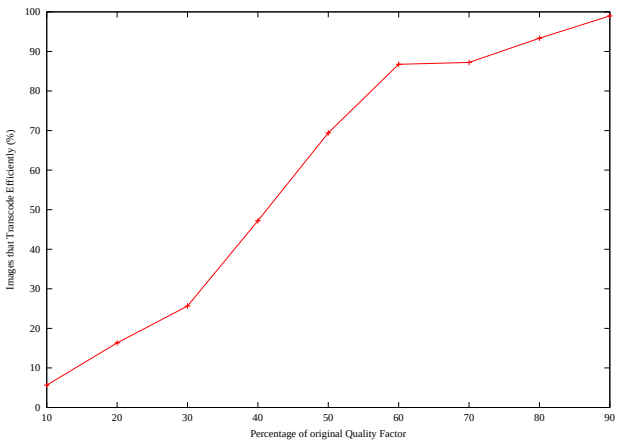
Web Server: We modified the Apache Server [6] to offer differentiated service by choosing a number of transcoded image variants. We used the Independent JPEG Group's [9] JPEG library, along with the transcoding algorithm described in [3] to generate variations of the images. We modified Apache server to keep track of the current bandwidth utilization for the different user classes. Since the request pattern and hence the bandwidth consumption is bursty, the server computes the bandwidth trend by averaging over the past 30 minute inter-



(a) Image JPEG Quality Distribution

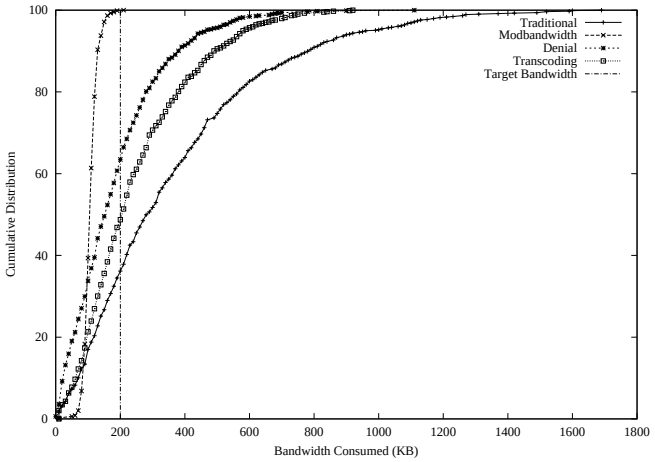


(b) Image File Size Distribution

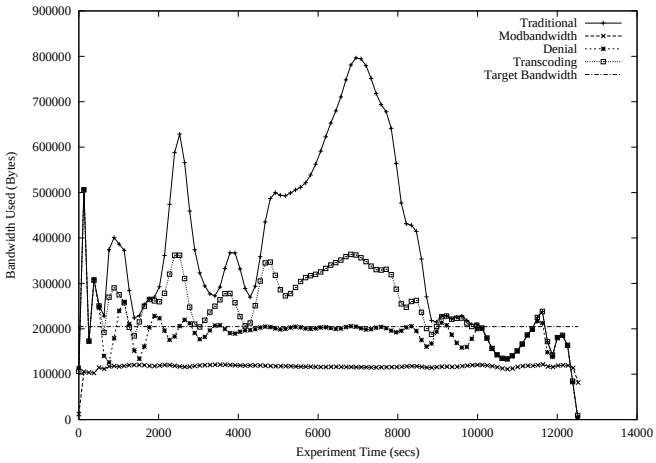


(c) Distribution of Efficient Images

Fig. 3. Workload Characteristics



(a) Bandwidth Consumption Distribution

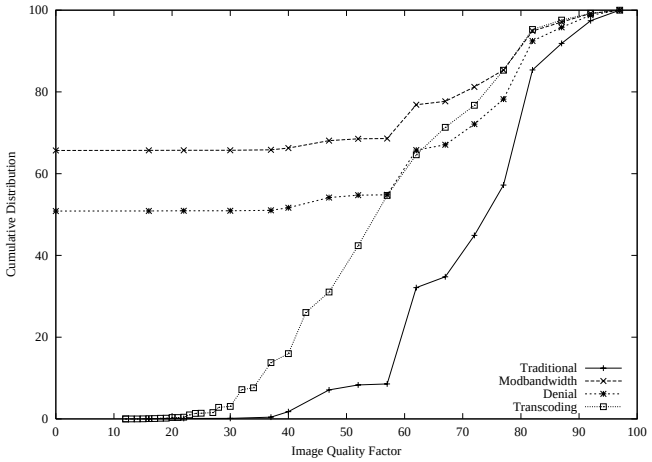


(b) Bandwidth consumed with time

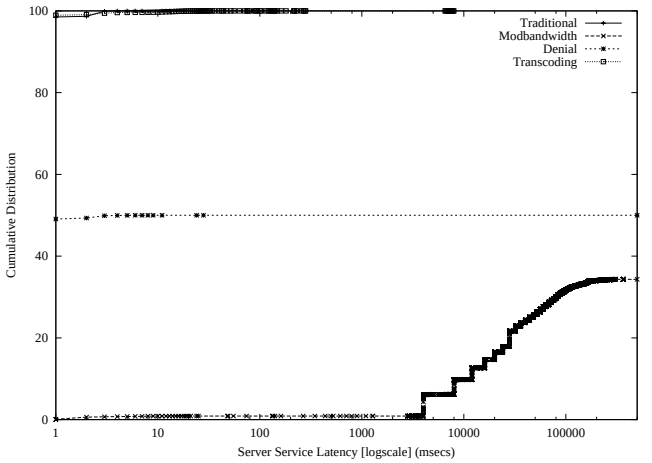
Fig. 4. Bandwidth Consumed (Target Bandwidth=200 KB/s)

val. The server implements the *Transcoding* mechanism as follows: If the average consumed bandwidth for the past half hour (measured every second) is more than the target bandwidth, the server serves proportionately lower quality variations of images. However when the consumed bandwidth exceeds twice the target bandwidth, the server denies further requests.

Client: We used http_load [16] to simulate accesses from clients using slow networks. http_load is a multi-processing http test client. We modified http_load so that it can compute the individual access latencies. We also modified http_load so that it can play back the client access traces generated from the NLANR proxy traces, as described in Section III-E.1.



(a) Image Quality Factor



(b) Client Access Latency

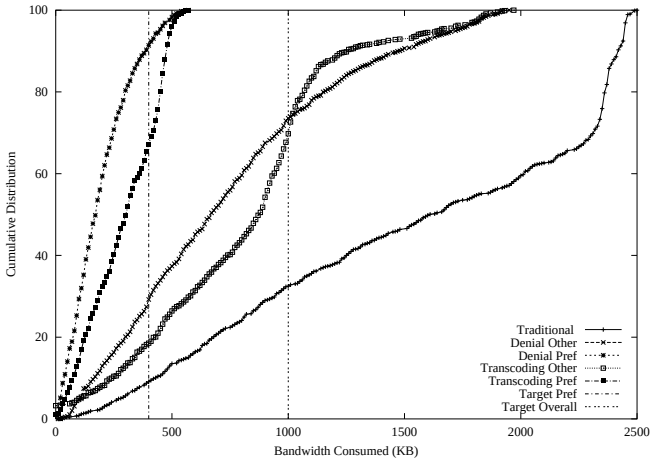
Fig. 5. Bandwidth Control (Target Bandwidth=200 KB/s)

IV. Results

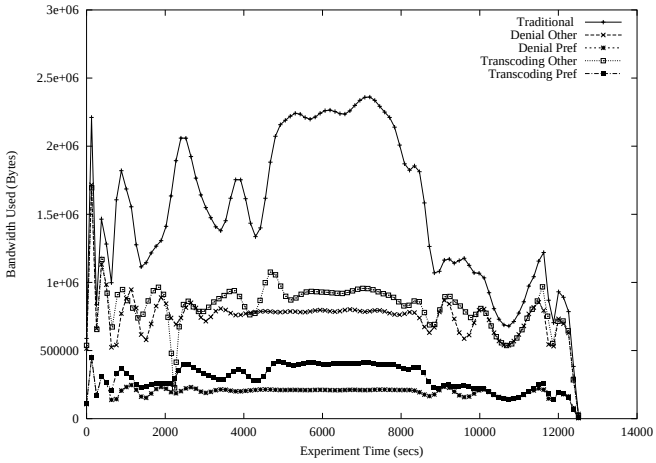
A. Bandwidth Control

First we analyze the ability of a web service to manage its bandwidth consumption using quality aware transcoding.

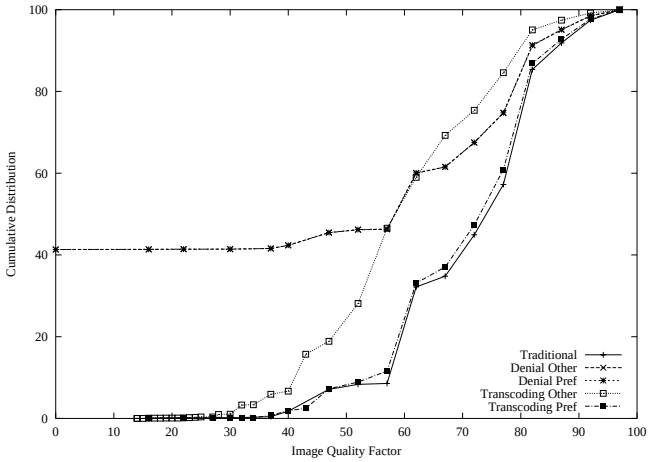
The raw access traces that we had collected from the NLANR proxies represent a fraction of the actual accesses to the web site. In order to simulate a realistic workload, we need to scale these traces to match the accesses to the actual server. Using the reach measures from Nielsen's Netratings [14], we estimated the average access rate to geocities.com at 154 accesses per second. We empirically measured that the *Modbandwidth* scheme cannot handle such loads because of the buildup of kernel buffer space as packets are delayed. Thus, for our experiments, we scale the number of raw accesses per time interval by a factor of thirty to generate a sufficiently de-



(a) Bandwidth Consumed



(b) Bandwidth Consumed with time



(c) Image Quality factor

Fig. 6. Differentiated Service (Preferred Clients=20%, Reserved Bandwidth=400 KB/s, Target Bandwidth=1 MB/s)

manding reference access stream. We measured the average access rate for our traces at 5.7 accesses/second. We simulated a web service that targets the bandwidth consumed to be 200 KB/s. We measured the average bandwidth demanded by this trace at 365 KB/s.

We plot the bandwidth consumed by the server, for the various service policies in Fig. 4. Fig. 4(a) plots the cumulative distribution of the bandwidth consumed and Fig. 4(b) plots the bandwidth consumption with experiment time (smoothed using a bezier function).

From 4(a), we note that *Traditional* overshoots the target bandwidth 65% of the time, while *Modbandwidth*, *Denial* and *Transcoding* overshoot the target bandwidth 0%, 35% and 45% of the time, respectively. Our *Transcoding* bandwidth management algorithm reacts to average consumed bandwidth within the past half hour. Hence, the algorithm misses sudden bandwidth spikes. We note that the percentage of egregious abuse of bandwidth (over 600 KB) is lower for *Transcoding* than with the *Traditional* policy.

From 4(a), we note that *Modbandwidth* and *Denial* provides the best control over bandwidth. However, as we demonstrate below, this control comes at the cost of driving client latencies to unacceptable levels. Recall that Apache controls bandwidth by delaying network bandwidth, increasing client latency and consuming additional kernel resources (mbufs). Fig. 4(b), also highlights the *Transcoding* scheme's inability to adequately respond to flash crowds. For example, the *Transcoding* scheme takes some time to respond to the flash crowds at time=2600 seconds. This delay forces the *Transcoding* to consume up to 400 KB/s. Also, when the target bandwidth is smaller than the requested bandwidth, (time=7000 seconds), the *Transcoding* scheme consumes as much as 400 KB/s because of the high bandwidth differential. This limitation is largely a result of the smoothing function we employ to filter out short term bursts in client accesses. We are currently investigating alternate functions that respond more quickly to sustained bursts while filtering short-term effects.

Next, we measure the performance of the system using the server latencies and Quality factor of the images served to the clients. Fig. 5(a) and Fig. 5(b) plot the Quality factor of the images and server latencies as cumulative distributions respectively. Denied requests are marked with a infinite service latency and Quality factor of 0.

From 5(a), we note that *Modbandwidth* denies over 65% of the requests. As the heavily loaded server tries to manage bandwidth by delaying network packets, the number of open network connections increases leading to service denial for newer client requests. *Denial* scheme denies about 50% of the requests. On the other hand, the *Transcoding* policy gracefully degrades the Quality factor for the images served.

From 5(b), we note that the service latencies are quite high for the *Modbandwidth* scheme; 90% of the requests take ten seconds or more to service. Images served by *Traditional*, *Denial* (those not denied) and *Transcoding* take less than a second. The x axis is plotted using a logarithmic scale.

We conclude that *Transcoding* provides useful bandwidth control by gracefully degrading image Quality factors. We note that *Transcoding* does not control bandwidth effectively in the presence of flash crowds and if there is a wide differential between target and requested bandwidth. *Denial* and *Modbandwidth* control their bandwidth consumption by (unacceptably) denying service for a large portion of the images.

B. Differentiated Web Service

Next, we analyze the ability of the bandwidth managements schemes to dynamically provide differentiated Web services to preferred clients. Since *Modbandwidth* performs with unacceptable latencies, we only compare the *Denial* and *Transcoding* schemes. We measure the performance of the *Traditional* scheme for reference.

The *Transcoding* scheme controls the consumed bandwidth for the different classes by proportionately reducing the image quality until the consumed bandwidth equals twice the target bandwidth, at which point further requests are denied. For preferred clients, the server reduces the image Quality factor of the images served at a rate that is proportional to the overall target bandwidth. For the rest of the clients, the server reduces the image Quality factor of the images served at a rate that is proportional to the left over bandwidth.

The *Denial* scheme manages bandwidth by serving a HTTP 503 error code when the consumed bandwidth overshoots the target bandwidth. The scheme does not deny service to preferred clients if the bandwidth consumption does not exceed the bandwidth allocated for preferred clients.

Since we do not perform experiments using the *Modbandwdith* scheme, we are able to scale the number of raw accesses per time interval by a factor of 150 to generate the reference access stream for this set of experiments. We measured the average access rate for our traces at 28.6 accesses/second. We simulated a web service that targets the bandwidth consumed to be 1 MB/s. We measured the average bandwidth demanded by this trace at 1.5 MB/s. To provide differentiated service, we configured the *Denial* and *Transcoding* schemes to allocate 40% of their available bandwidth for the preferred clients. We configured 20% of the clients to be preferred clients. The test client notifies the server of the request class using custom HTTP headers.

The results are plotted in Fig. 6. Fig. 6(a), plots the bandwidth consumed by the server as a cumulative distribution while Fig. 6(b) plots the bandwidth consumed with experiment time. Fig. 6(c) plots the cumulative distribution of the average Quality factor of the images served. Throughout the experiments, we measured the service latency to be under a second for images served (i.e., when not denied) for the various user classes.

From Fig. 6(a), we note that the *Denial* and *Transcoding* provide bandwidth control for the different client classes. For the preferred clients in the *Denial* and *Transcoding* schemes, 90% and 65% of the preferred clients consume less than 400

KB/s.

Fig. 6(b), shows the results with experiment time for reference. Since the target bandwidth is closer to the maximum bandwidth, the system has better latitude in managing its bandwidth and hence better handles the heavy load (at time=4500 seconds).

From Fig. 6(c), we note that *Denial* scheme denies service for 40% of the preferred and general clients. With increased server load, the *Denial* scheme does not have latitude in managing differentiated service. We speculate that a better alternative for providing differentiated service would be to completely deny service to non-preferred clients.

However, *Transcoding* provides graceful degradation of image Quality factors with the preferred clients served at Quality factors that closely follow the original images. Non-preferred clients are served at a lower image Quality factor.

Hence we conclude that for a service offering differentiated Quality of service, *Transcoding* provides images of better Quality factors to the preferred clients while degrading gracefully for non-preferred clients. *Denial* could not provide differentiated QoS without completely denying accesses to non-preferred clients.

V. Related Work

A. Network Level QoS techniques

A number of research efforts focus on providing differentiated service in the networking infrastructure. For example, the IETF working groups on Integrated Services [17], [18] and Differentiated Services [19] have identified a number of issues in providing differentiated service at the network level. A number of systems [20], [21], [22] have used network level techniques to provide differentiated services for the Internet. Vogel et al. [23] present a survey of techniques used to provide quality of service within distributed multimedia systems.

B. System Level QoS techniques

At the system level, a number of systems attempt to provide differentiated quality of service for the web using priority based schemes. The fundamental problem faced by a priority based scheme is that lowering the delivery priority increases the access latencies for multimedia objects (as multimedia objects tend to be large). Traditional human factors research [24] has shown that the response time for accessing a resource should be in the 1 to 10 second range for information to be useful. If the response time is longer than this range, the users tend to lose interest and go on to other things. In a competitive world, any inaccuracy in prioritizing traffic can exacerbate the access latencies, potentially turning away customers who were deemed less important.

Mod_bandwidth [25] is an Apache bandwidth management module that enforces per directory bandwidth limits based on a number of configurable parameters. Bandwidth usage is limited by delaying packet delivery, with the side effect of increasing client access latency. Because of the large sizes of multimedia objects, per-packet delays may have an exaggerated impact on latency. Almeida et al. [26] describe a system that uses a priority-based system to schedule the priority web services to offer differentiated levels of service for web hosting services. Pandey et al. [27] describe a web server that allows for setting priorities among page requests and allocating server resources to enforce QoS constraints. Banga et al. [28] describe a system that uses resource containers for providing services based on the end-to-end resource container. Eggert et al. [29] describe a web service that uses mechanisms such as limiting process pool size, lowering process priorities and limiting transmission rate to provide different levels of service. Banâtre et al. [30] describe a system that uses profile-based a predictive pre-fetching policy to improve response times for related profiles. These priority based systems increase the access latency for lower priority clients at the expense of higher priority clients, potentially leading to unacceptable delays for low priority clients.

Commercial systems such as WebQoS [31] from HP provides quality of service on the Web by using priority levels to determine admission priority and performance-level. WebQoS uses parameters such as source IP address, destination IP address, URL, port number, hostname and IP type-of-service to classify requests. The system uses these priorities in controlling the allocation of CPU and disk resources. Higher priority requests are sent to servers running in separate ports that operate under different system priorities. The system uses priorities to delay or deny service to lower priority clients. Though this policy leads to predictable service for the preferred clients, the lower priority clients can be turned away unnecessarily.

C. Application Level QoS techniques

Edell et al. describe an alternative ISP system called INDEX [32] that offers differentiated quality of service. The users dynamically choose their level of network quality based on the resource cost. Our work adds another dimension to the users choice by allowing the user to select a lower quality multimedia object on a slower (cheaper) network in order to improve the access latency.

Traditionally Web services have used mirroring as a means for addressing the problem of exploding web traffic by replicating objects closer to the end user. Vahdat et al. describe an Active Name system [33] that solves the problem of locating the closest mirrors by downloading computations into the network infra-structure. A similar approach was used by systems such as Active Caching [34] and Active Networks [35] to offload computations into the network infrastructure to reduce the network load on the end servers.

Commercial systems such as Footprint [36] and FreeFlow [37] enable web sites to migrate their contents and route the user to the closest replica. Pricing for these services are based on reserved and peak aggregate bandwidth.

Our work compliments these systems by allowing a web service to manage its expensive network bandwidth consumption

regardless of whether the object was served from the origin server or from replicas.

VI. Conclusions and Future Work

In this paper, we explore a web service that uses informed transcoding to manage its bandwidth consumption. We show that transcoding can allow the server to manage its bandwidth without adding excessive latency or denying service. Transcoding also allows the web server to provide differentiated service by allocating its bandwidth for different usage classes. We show that the service degrades the quality gracefully for different user classes, while still managing overall consumed bandwidth effectively. We are currently investigating techniques to allow web designers to specify the relative importance of various multimedia components of web pages, e.g., to specify the relative importance of objects allowing the server to choose the transcoding level intelligently on a per-object basis.

Acknowledgments

This work was supported in part by a graduate fellowship from North Carolina Networking Initiative (NCNI) and equipment grants from Intel Corporation and the National Science Foundation (CDA-95-12356). We would like to thank IBM Research for various resources used in producing this paper. We would also like to thank Darrell Anderson for his assistance in configuring the FreeBSD kernel.

References

[1] "Khera Communications Inc.," www.kciLink.com.

[2] Antonio Ortega, Fabio Carignano, Serge Ayer, and Martin Vetterli, "Soft caching: Web cache management techniques for images," in *IEEE Signal Processing Society 1997 Workshop on Multimedia Signal Processing*, Princeton NJ, Jun 1997.

[3] Surendar Chandra and Carla Schlatter Ellis, "JPEG Compression Metric as a Quality Aware Image Transcoding," in *2nd Symposium on Internet Technologies and Systems*, Boulder, CO, October 1999, USENIX.

[4] William B. Pennebaker and Joan L. Mitchell, *JPEG - Still Image Data Compression Standard*, Van Nostrand ReinHold, NY, 1993.

[5] Surendar Chandra, Ashish Gehani, Carla Schlatter Ellis, and Amin Vahdat, "Transcoding characteristics of web images," Tech. Rep. CS–1999–17, Department of Computer Science, Duke University, November 1999, (submitted to WWW9).

[6] Apache Group, "Apache web server version 1.3.6," www.apache.org.

[7] Armando Fox, Steven D. Gribble, Eric A. Brewer, and Elan Amir, "Adapting to network and client variability via on-demand dynamic distillation," *ACM SIGPLAN Notices*, vol. 31, no. 9, pp. 160–170, Sept. 1996, Co-published as SIGOPS Operating Systems Review **30**(5), December 1996, and as SIGARCH Computer Architecture News, **24**(special issue), October 1996.

[8] Eric Hamilton, *JPEG File Interchange Format - Version 1.02*, C-Cube Microsystems, 1778 McCarthy Blvd, Milpitas, CA 95035, September 1992.

[9] Tom Lane, Philip Gladstone, Luis Ortiz, Jim Boucher, Lee Crocker, Julian Minguillon, George Phillips, Davide Rossi, and Ge' Weijers, "The independent jpeg group's jpeg software release 6b," ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz.

[10] Adrian M. Ford, *Relations between Image Quality and Still Image Compression*, Ph.D. thesis, University of Westminster, May 1997.

[11] R. E. Jacobson, A. M. Ford, and G. G. Attridge, "Evaluation of the effects of compression on the quality of images on a soft display," in *Proc. of SPIE: Human Vision and Electronic Imaging II*, San Jose, CA, Feb 1997.

[12] Tim Berners-Lee, R. T. Fielding, H. Frystyk Nielsen, J. Gettys, and J. Mogul, *Hypertext Transfer Protocol – HTTP/1.1*, January 1997.

[13] The National Laboratory for Applied Network Research, "A distributed testbed for national information provisioning," http://ircache.nlanr.net/.

[14] "Nielsen net rating service," http://nielsen-netratings.com/, March 1999.

[15] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, November 1997.

[16] Jef Poskanzer, "Http load - multiprocessing http test client," www.acme.com/software/http_load/, 1998.

[17] "Integrated services working group," www.ietf.org/html.charters/intserv-charter.html.

[18] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," RFC 2212, September 1997.

[19] Yoram Bernet, James Binder, Steven Blake, Mark Carlson, Brian E. Carpenter, Srinivasan Keshav, Elwyn Davies, Borje Ohlman, Dinesh Verma, Zheng Wang, and Walter Weiss, "A framework for differentiated services," draft-ietf-diffserv-framework-02.txt, February 1999.

[20] Farooq M. Anjum and Leandros Tassiulas, "Fair bandwidth sharing among adaptive and non-adaptive flows in the internet," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. March 1999, pp. 1412–1420, IEEE.

[21] Laurent Massoulie and James Roberts, "Bandwidth sharing: Objectives and algorithms," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. March 1999, pp. 1395–1403, IEEE.

[22] Martin May, Jean-Chrysostome Bolot, Alain Jean-Marie, and Christophe Diot, "Simple performance models of differentiated service schemes for the internet," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. March 1999, pp. 1385–1394, IEEE.

[23] Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann, and Jan Gecsei, "Distributed multimedia and QOS: A survey," *IEEE Multimedia*, vol. 2, no. 2, pp. 10–19, Summer 1995.

[24] Jakob Nielsen, *Usability Engineering*, Academic Press, Boston, MA, 1993, (hardcover), 0-12-518406-9 (paperback).

[25] Yann Stettler, "Bandwidth management module for apache webserver," www.cohprog.com, Mar 1997.

[26] Jussara Almeida, Mihaela Dabu, Anand Manikutty, and Pei Cao, "Providing differentiated levels of service in web content hosting," in *Proceedings of the Workshop on Internet Server Performance*, Madison, Wisconsin, June 1998.

[27] Raju Pandey, J. Fritz Barnes, and Ronald Olsson, "Supporting quality of service in HTTP servers," in *PODC: 17th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 1998, pp. 247–256.

[28] Gaurav Banga, Peter Druschel, and Jeffrey C. Mogul, "Resource containers: A new facility for resource management in server systems," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. USENIX Association, February 1999, pp. 45–58, ACM SIGOPS.

[29] Lars Eggert and John Heidemann, "Application-level differentiated services for web servers," *World-Wide Web Journal*, vol. 2, no. 3, pp. 133–142, August 1999.

[30] Michel Banâtre, Valérie Issarny, Frédéric Leleu, and Boris Charpiot, "Providing quality of service over the web: A newspaper-based approach," in *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, CA, April 1997.

[31] "WebQoS Version 2," www.hp.com/go/webqos.

[32] Richard J. Edell and Pravin P. Varaiya, "Providing internet access: What we learn from the INDEX trial," Project Report 99-010W, University of California, Berkeley, April 1999.

[33] Amin Vahdat, Michael Dahlin, Thomas Anderson, and Amit Aggarwal, "Active names: Programmable location and transport of wide-area resources," in *2nd Symposium on Internet Technologies and Systems*, Boulder, CO, October 1999, USENIX.

[34] Pei Cao, Jin Zhang, and Kevin Beach, "Active cache: Caching dynamic contents on the web," in *Proceedings of Middleware '98*, 1998.

[35] Ulana Legedza, David J. Wetherall, and John Guttag, "Improving the performance of distributed applications using active networks," in *IEEE Infocom*, March 1998.

[36] Sandpiper Networks Inc., "Footprint adaptive content distribution service," www.sandpiper.net.

[37] Akamai Technologies Inc., "FreeFlow content distribution service," www.akamai.com.