Application-Level 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 content is constrained by available network bandwidth. 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 using application-specific characteristics of web services. 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 our earlier work on characterizing quality versus size tradeoffs in transcoding JPEG images to supply more information for determining 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 using 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 good quality of information content for all clients. We also show that it is possible to customize multimedia objects to the highly variable network conditions experienced by mobile clients in order to provide acceptable quality and latency depending on the networks used in accessing the service. We show that policies that aggressively transcode the larger images can produce images with Quality Factor values that closely follow the un-transcoded base case while still saving as much as 150 KB. A transcoding policy that has knowledge of the characteristics of the link to the client can avoid as many as 40% of (unnecessary) transcodings.

Keywords

Quality Aware Transcoding, Differentiated Web Service, Quality of Service

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). The NLANR proxy traces were made possible by the National Science Foundation (grants NCR-9616602 and NCR-9521745) and the National Laboratory for Applied Network Research

I. INTRODUCTION

THE web is emerging as the primary data dissemination mechanism for a variety of applications. Recently, we have seen the increasing popularity of Internet services such as e-commerce and web hosting. One goal of a web service is to provide low latency access to its resources. However, this goal is constrained by available network bandwidth. The web service needs to serve as many users as possible at sufficiently attractive levels of quality and latency to gain and retain their business. A web server experiencing heavy network load is not able to provide satisfactory service to all its clients. Web sites also need to maintain QoS for different client classes in order to retain preferred, paying customers as well as attract potential customers. Further, a client accessing a web service using a slow link (e.g. wireless) is constrained by the low bandwidth available to the service. Both baseline network characteristics as well as prevailing congestion dictates the QoS experienced by the end users.

The problem of slow, expensive and congested networks is compounded by the large size of multimedia objects that are becoming such a prevalent part of web content. By some estimates [1], about 77% of the 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. In general, there is a huge mismatch between the rich multimedia content available on the web and the characteristics of network technologies that are used to access them. For instance, mobile users desire to minimize access time and cost while servers do not want large multimedia objects to affect their ability to serve preferred clients.

In the past, web sites have used ad-hoc solutions to deal with peak loads. For example, when significant news stories break, news sites such as MSNBC and CNN export a lightweight version of their content with little or no graphics to conserve bandwidth. Caching in the network infrastructure at web proxies is another traditional technique used 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). Also, web sites expressly disable caching in order to maintain control over information about access pattern and hit counts.

One approach to this problem is to provide differentiated service at the web server. Differentiated service allows servers to match object sizes with the available network bandwidth by providing different versions of the same object to different clients. Differentiated service also allows a service to dynamically partition its network resources to its preferred clients, allowing the system to provide better service for certain customers based on their status. Servers can use network parameters such as client IP addresses, browser parameters such as cookies as well as the client transaction state (e.g. client is about to approve the purchase) to identify the various client classes.

In this work, we allocate critical network resources dynamically at the application level in response to client access patterns by tailoring the content itself. The feasibility of such a differentiated service scheme depends on the availability of a range of variations

for the content so that the server can choose the correct variation given a request and current network conditions. While the content provider can manually provide a number of different variations for use by the system, an automatic technique is preferable to allow the system to dynamically adapt to variability in network performance.

Our approach to providing differentiated QoS is through transcoding, a transformation that is used to convert a multimedia object from one form to another (frequently trading off object fidelity for size). Transcoding operations are often performed to fit an object to the characteristics of the display device. Sample image transcodings include transformations to thumbnails, gray-scale, progressive formats as well transcodings to textual information. By their very nature, multimedia objects are amenable to soft access through a quality-versus-size tradeoff. Our focus is on transcoding to reduce bandwidth requirements on the server. Transcoding allows web services 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.

For transcoding to be useful in providing differentiated service, we need to understand its tradeoff characteristics: the information quality loss, the computational overhead required in computing the transcoding and the potential space benefits. Earlier work [1] showed that 67% of the Internet web data are transferred for images. In our earlier work [2], we analyzed the images accessed in the Internet and their transcoding characteristics. We showed that Internet image data is evenly distributed between GIF and JPEG images, with JPEG offering the most potential benefits. We also established the characteristics of popular image transcoding operations. Our results allow us to choose the appropriate transcoding techniques for the most important classes of Internet images.

To illustrate the potential of using transcoding to dynamically customize multimedia images, we previously characterized the tradeoffs of a transcoding that changes the JPEG [3] compression metric [4]. Prior to our work, other systems typically transcoded images to ad-hoc Quality Factor values, which can potentially lead to an increase in the image size for certain images. Other systems countered this counter-productive behavior by (unnecessarily) transcoding all images to a conservatively low Quality Factor value. In our work, we developed techniques to measure the initial Quality Factor of an image, allowing us to explore *quality-aware* policies that can transcode images to a fraction of their original image Quality Factor or to a target size. We also characterized the computational costs and the space benefits possible with a particular transcoding. Such characterization allows us to explore policies that use information about the link characteristics to dynamically decide if transcoding will be worth the effort for a particular request. Such characterization enables the system to make an informed decision in choosing the appropriate version of an object to transmit to an end user.

In this paper, we explore the potential benefits of providing differentiated QoS for different classes of users through informed transcoding for better utilization of limited network resources. For our experiments, we modify the Apache web server [5], one of the most popular Internet web servers. We use realistic workloads gleaned from popular web sites and topical web proxy access

traces to drive our system. We use the JPEG compression metric as the informed transcoding technique. While we restrict our efforts to this single metric, our techniques are equally valid for any transcoding with well-understood tradeoff characteristics.

The principal contribution of this work is the demonstration that it is possible to use informed transcoding techniques to provide 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 [6]. We also show how a web server can dynamically serve different versions of multimedia objects to clients accessing the web, based on network performance, to provide tolerable latency and a "satisfying web experience" [7].

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

II. Related Work

A. Internet Image Transcoding Characteristics

It is important to understand the nature of typical images in the current Internet and their transcoding characteristics in order to choose the appropriate transcoding operation. In our earlier work [2], we focussed our attention on transcodings that customized an image for file size savings. Such knowledge enables service providers to choose potential transcoding techniques that offer benefits for a wide variety of images. It also allows the service to avoid choosing transcoding techniques that might appear promising but are not effective for their target workload.

We analyzed images collected from the NCAR NLANR proxy access logs. We showed that 74.81% and 24.41% of the unique web images were GIF [8], [9] and JPEG [10] images respectively. Other image formats, including PNG, make up the rest of the (0.78%) requests. Because of their dominance, we further analyze GIF and JPEG images. We showed that most of the GIF images are small; about 80% of the GIF images are smaller than 6 KB. About 45% of these GIF images appear to be bullets, icons, lines or banners. We argued that, on average, JPEG images are larger than GIF images; 40% of the JPEG images are larger than 6 KB. About 30% of the small JPEG images appear to be bullets, icons, lines or banners. About 35% of JPEG images (78.3% of JPEG data) and 10% of GIF images (45.1% of GIF data) are large images that cannot be categorized as bullets, icons, lines or banners. These image characteristics have implications for transcoding techniques.

Our evaluations show that for JPEG images, the JPEG compression metric and a transcoding that reduces the spatial geometry are productive transcoding operations. Unless proper transcoding parameters are chosen, traditional transcoding techniques such

as thumbnailing of GIF images have the potential of actually increasing the output image file size. For a transcoding that reduces the image by a factor of 2 and 4 along each axis, 40% and 2% of the GIF images transcode to a size that is larger than the original image file size, respectively. There is significant opportunity for sophisticated transcoding of JPEG images. The JPEG compression metric loses visually imperceptible information first. Hence, it is a good transcoding that reduces image file size, sacrificing as little visual information as possible.

B. Quality Aware Transcoding

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. Han et al. [11] present an analytical framework for determining whether to transcode and how much to transcode an image. However, their quantification does not take the image information quality into account and hence the information quality loss is not quantified.

Our earlier effort into quality aware transcoding [4] is the enabling technology for this research. We characterized the tradeoffs inherent in a transcoding that changed the JPEG compression metric (also referred to as the JPEG Quality Factor). Reconstructing the original Quality Factor used to produce the image is necessary so loss in quality becomes meaningful. Using the quantization tables stored in the JFIF [10] headers, we developed an algorithm to predict the Independent JPEG Group's (IJG) [12] 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 [13], [14].

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 (this count is constant for a given JPEG image). 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 will lose 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 efficiently at a significantly better percentage than the base case of all the images.

C. Differentiated Services Schemes

A number of research efforts focus on providing differentiated service in the networking infrastructure. For example, the IETF working groups on Integrated Services [15], [16] and Differentiated Services [17] have identified a number of issues in providing differentiated service at the network level. A number of systems [18], [19], [20] have used network level techniques to provide differentiated services for the Internet. Vogel et al. [21] present a survey of techniques used to provide QoS within distributed multimedia systems.

At the system level, a number of systems [22], [23], [24], [25], [26], [27] attempt to provide differentiated QoS 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 (which tend to be large). Traditional human factors research [28] 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.

At the application level, Abdelzaher et al. [29], [30] describe a web server that manages its consumed resources by adapting the web content. Static variations of the web content trees are available to the web server and the server utilizes the content trees with successively smaller objects to manage its resource consumption. Our work complements this system by providing a methodology to dynamically generate the content variations with well quantified Quality Factor loss. Edell et al. [31] describe an ISP system that offers differentiated QoS by letting the users dynamically choose their level of network service based on the resource cost. Our work adds another dimension to the user's choice by allowing the user to select a lower quality multimedia object to improve the access latency. Commercial systems such as Footprint [32] and FreeFlow [33] enable web sites to migrate/replicate their contents and route the user to the closest replica. Our work complements 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.

D. Transcoding Network Proxies

A number of systems such as [34], [35], [36], [37], [38], [39], [40], [1], [41], [42], [43], [44], [45], [46] have used various forms of compression and transcoding operations to improve web access from slow networks. However, those systems do not use an informed transcoding technique that can quantify the information loss from the chosen transcoding operation and hence perform ad hoc transcoding without an explicit understanding of the tradeoffs and potential gains. Noble et al. [47] describe a system wherein the bottleneck bandwidth need not be within the last hop to the client. Their results complement our work by helping the proxies make better informed decisions.

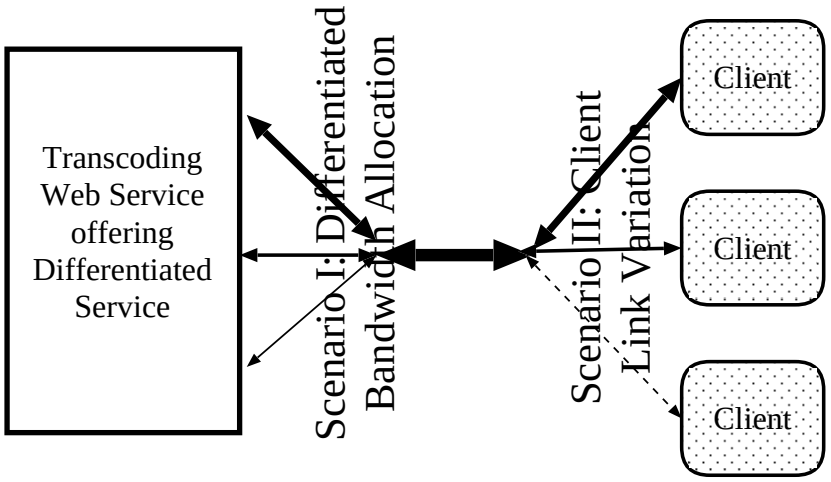


Fig. 1. System Architecture

III. Experiment Objectives and Design

A. System Architecture

The system that we envision is illustrated in Fig. 1. The web server uses transcoding to provide differentiated service. In scenario I, transcoding is performed by the web server to manage the network bandwidth available at the server. Throughout this paper, we will refer to this scenario as the *server-oriented scenario*. In scenario II, transcoding is performed to fit the multimedia image to the network bandwidth of a mobile client using a slow and expensive link. Throughout this paper, we will refer to this scenario as the *client-oriented scenario*.

The transcoding operation adds a computational overhead, as quantified in [4]. In this study we are mostly concerned with the effect on the network load. Hence, the transcoding required to provide differentiated service for images is performed once and the results of the transcoding are cached for reuse.

B. 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 to the rest of the wide-area network?
- For a web server offering differentiated service, can quality aware transcoding allow the web service to provide differentiated service for preferred and ordinary clients?
- For a web server offering differentiated service to clients accessing the service using slow (wireless) links, what is the delivered performance, as defined by image quality and bandwidth savings, to end users?

C. Performance Measures

Since transcoding trades-off image quality for size, the Quality Factor of images served gives an indication of the quality tradeoff. The overall goal is to maintain as much information quality as possible, within the constraints of the available network bandwidth. Each of the specific scenarios warrant additional performance measures.

In the server-oriented scenario, the chief constraint to the ability to serve many users is the limited network bandwidth available to the wide area network. Bandwidth control is the primary objective for our research. The goal is to maintain the bandwidth consumed within pre-defined levels, regardless of the demand. The client experience not only depends on the information quality factor, but also on the latency of accessing the images. In such an operating environment, *Bandwidth Consumed* and *Client Latency* additionally capture the performance of a web server. Using results from human factors research, we use ten seconds as the threshold on acceptable latency.

On the other hand, for a client-oriented scenario the type of differentiated service our server hopes to offer is to provide useful quality images adapted to the available network bandwidth. The chief constraint to their ability to serve users is the limited and variable network bandwidth available to the clients, often within the last hop. In such an operating environment, the measure *Number of bytes not sent* additionally captures the network bandwidth savings. For expensive networks, this measure directly translates to cost savings in accessing the web information. For slow networks, this measure translates to reduced request latency due to transmission.

D. Web Service Policies

In the server-oriented scenario, we perform experiments to 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:* Then, we consider the performance of a bandwidth management scheme that 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 [48] error code “503: Service Unavailable”. The scheme does not deny service to preferred clients if the bandwidth consumption does not exceed the bandwidth allocated for preferred clients.

- *Transcoding:* Finally, we study the performance of a web service that offers differentiated service by transcoding images 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. 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 leftover bandwidth. When the consumed bandwidth equals twice the target bandwidth, subsequent requests are denied. A goal in image transcoding is to ensure 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 [4].

In the client-oriented scenario, we perform experiments to explore the following transcoding policy alternatives:

- *Simple:* Following current practice, we transcode the JPEG images to ad hoc JPEG Quality Factor values of 25, 50 and 75.
- *Relative quality:* We transcode the JPEG images to a percentage of the original JPEG Quality Factor values (25%, 50% and 75%).
- *Target size:* We transcode the JPEG images to Quality Factor values that are estimated to achieve a target image size of 4 KB, 10 KB and 20 KB. Our heuristic assumes that all JPEG images lose at least as much in image size as they lose in image quality. We have experimentally verified that this linear assumption is valid for about 80% of the images. Images that are smaller than this target size are not transcoded. This policy aggressively transcodes large files.
- *Informed:* We use informed transcoding to only transcode *efficient* images. We use the following algorithm to decide if it is worth transcoding an image: If the estimated time to transmit the original image, given the current estimate of network bandwidth available, is less than the estimated time to compute the transcoding on the server and the time to send the transcoded image then the original image should always be sent. Otherwise, transcoding is considered worthwhile if the difference in the above estimates exceeds a threshold, the original image quality is exceptionally high, or the percentage of low frequency components is high. We note that for our experiments on a 450 MHz Pentium III, the basic computational block described in [4] takes 7 $\mu secs$. For example, transcoding a typical 480×480 full color image (with 3 color components YCbCr and 2×1 scaling for chrominance values) corresponds to 7200 computations or 50 msec on this 450 MHz Pentium III processor.

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.

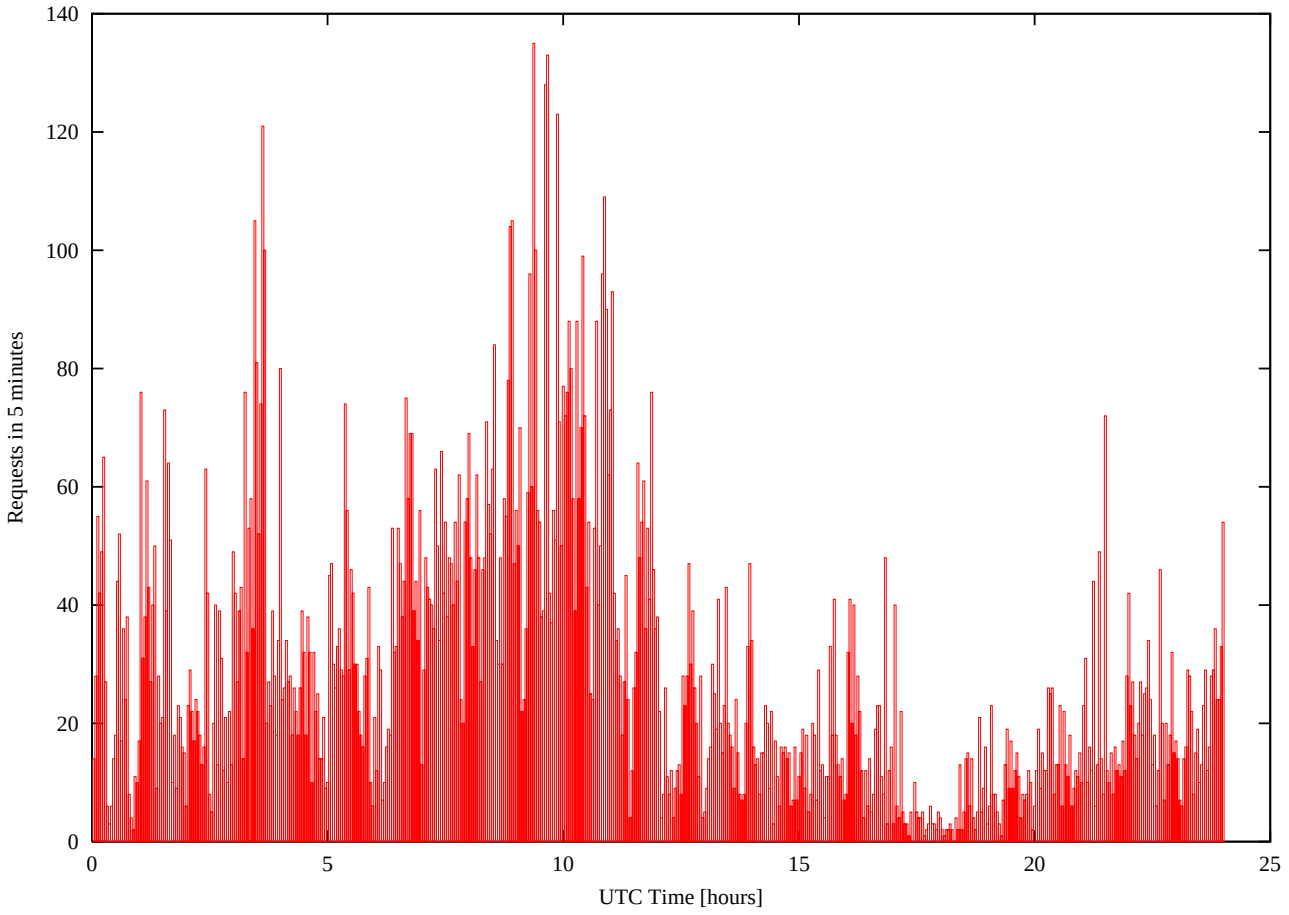


Fig. 2. Accesses to Geocities.com via IRCache @ Boulder, CO (Mar 23, 1999)

E.1 Server-Oriented Scenario

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 generate the access trace to the web service. Then, we need to choose 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.

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 [49] proxy caches. For our experiments, we analyzed 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 Net-rankings [50]. 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. Other studies [51] have observed self-similar behavior wherein the accesses look bursty and variable across all time scales. Over a 24 hour period, these accesses show an overall trend of higher numbers of accesses during evening hours and lower numbers during early morning hours. Earlier work [52] identified similar diurnal variations. [51] suggest aggregation as a mechanism to smooth the variations. We use an aggregation interval of 30 minutes to smooth the accesses.

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. For our experiments, we scale our access trace to approximately 12,500 seconds (3.5 hours). 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 Net-ratings [50], 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 in Section IV-A.1, we scale the number of raw accesses per time interval by a factor of thirty to generate a sufficiently demanding reference access stream. We measured the average access rate for our traces at 5.7 accesses/second.

However, since we do not perform experiments using the *Modbandwdith* scheme for a scheme offering differentiated service to different client classes (Section IV-A.2), we were 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 notified the server of the request class using custom HTTP headers.

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 be 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 original JPEG Quality Factor and the image size 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 transcoding variations possible 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. The results are plotted in Fig. 3(c).

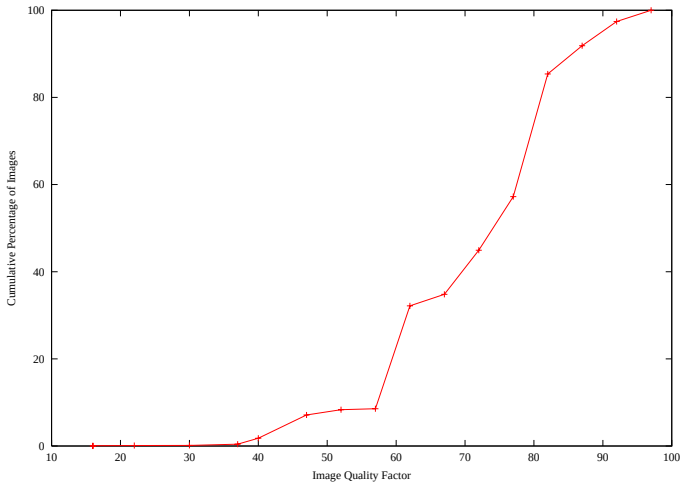
From Fig. 3(c), we note that there is a significant drop in the percentage of images that transcode efficiently when the images are transcoded to a Quality Factor of 30% or less of the original image Quality Factor. Hence, for our experiments, we transcode the original images to Quality Factor values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

The next workload parameter we consider involves developing ways to generate accesses from different client classes. For our experiments, we use http_load [53] to generate client accesses. http_load is a multi-processing http test client. We modified http_load to generate a configurable percentage of requests from different client classes. The http_load informed the server about the class that a request belongs to using custom HTTP headers.

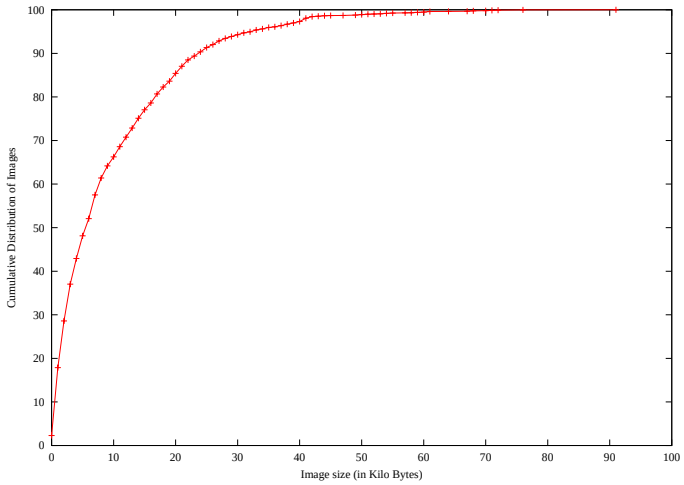
E.2 Client-Oriented Scenario

We first try a sample of images available on the Internet that are not specific to a single site. We analyzed the proxy traces collected on March 23, 1999 from the NLANR proxy at NCAR and NCSA. We utilize 13711 JPEG images (totaling 205 MB), and 11771 JPEG images (totaling 171 MB) downloaded from the NCAR and NCSA cache access logs respectively. We also consider synthetic access patterns consisting of requests for images available from typical web sites. We utilize 6217 JPEG images (totaling 70 MB), 4650 JPEG images (totaling 480 MB) and 1248 JPEG images (totaling 18 MB) from Cnn.com (News site) [54], Photo.net (Image site) [55] and Starwars.com (Commerce site) [56] respectively.

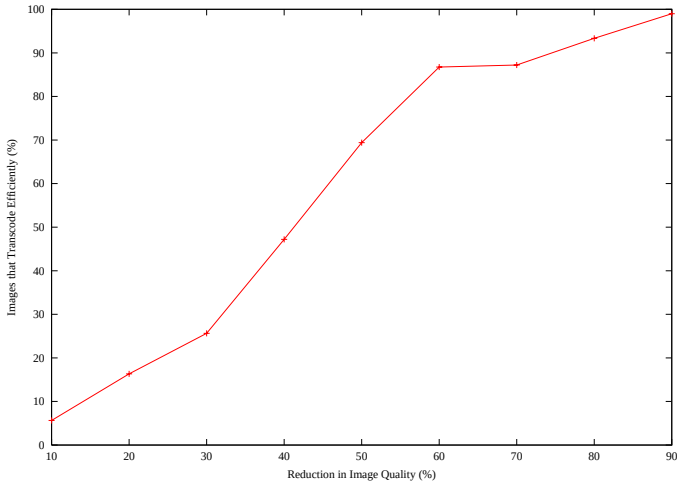
For the JPEG images in the various workloads, we plot the cumulative distribution of the image size and the original JPEG Quality Factor in Fig. 4. From Fig. 4(a), we note that images from PhotoNet are of high quality. From Fig. 4(b), we note that size distributions are similar for Cnn and Starwars with PhotoNet as the exception. The images are predominantly small, 80% of the images being smaller than 10 KB. As expected, PhotoNet has a significant proportion of images that are greater than 100 KB. PhotoNet offers images in three different sizes, thumbnails, regular sized and large images. Hence we note that the cumulative



(a) Image JPEG Quality Distribution

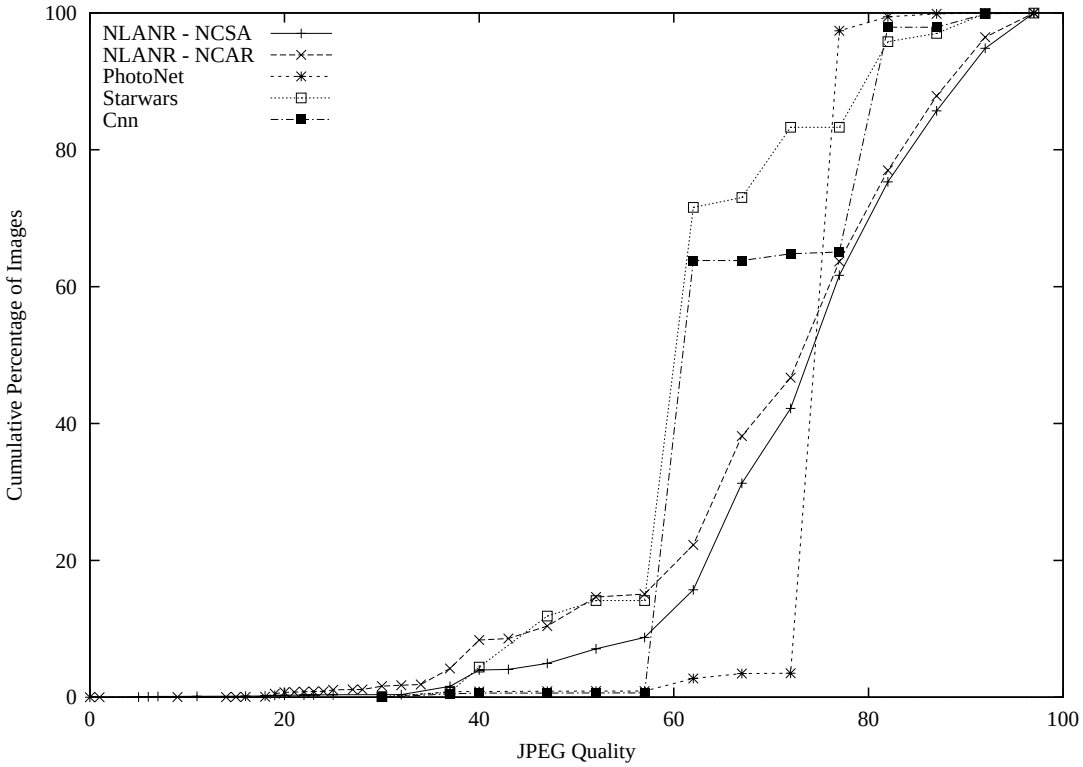


(b) Image File Size Distribution

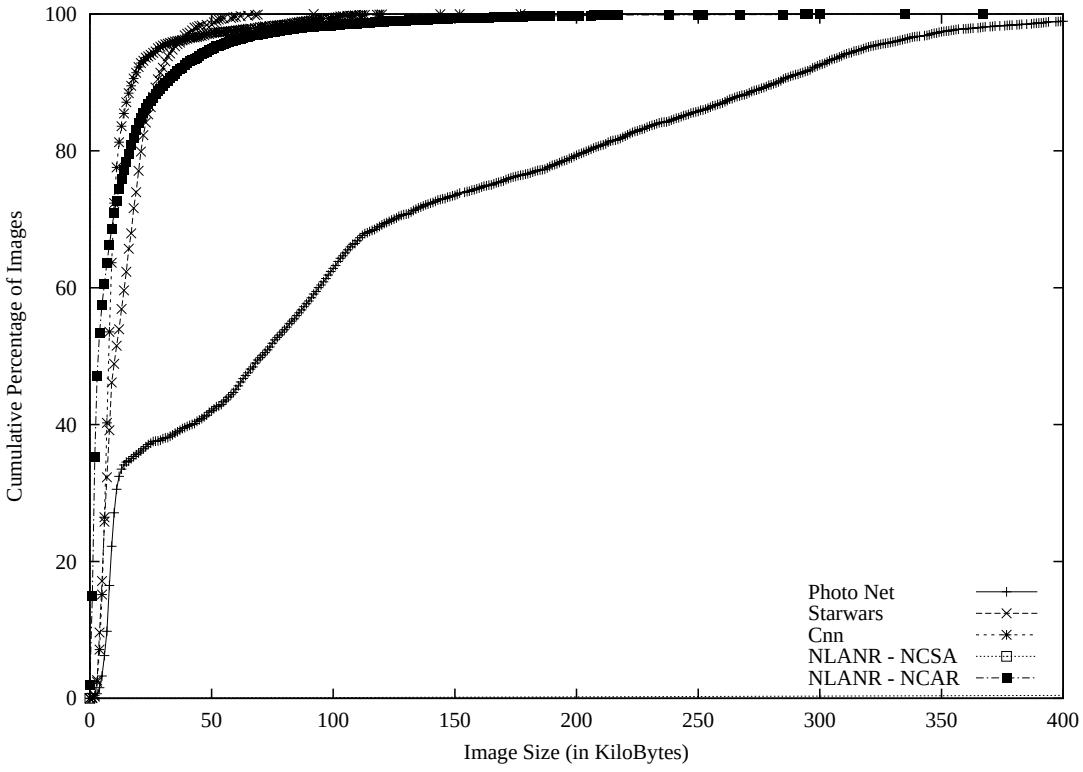


(c) Distribution of Efficient Images

Fig. 3. Server Oriented Scenario: Workload Characteristics



(a) Cumulative distribution of image JPEG Quality Factor



(b) Cumulative distribution of image File Size Distribution

Fig. 4. Client Oriented Scenario: Workload Characteristics

distribution curve has knees at 10 KB and 100 KB.

In the interest of space and based on the image characteristics, we use the NCSA and PhotoNet image collections for further discussions. NCSA is an example of a workload with many small files and PhotoNet represents sites with large, high quality images.

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.

We modified the Apache Server version 1.3.6 [5] to transcode JPEG images according to the various transcoding policies. For transcoding JPEG images, we used the IJG JPEG library. We used transcoding algorithms described in [4] to measure the initial JPEG Quality Factor of an image, as well as to transcode images by operating in the frequency domain, avoiding the need to uncompress the image completely. Throughout our experiments, we turned off HTTP persistent connections so that the server serves every image using a new connection. We used http_load to simulate accesses from clients using slow networks. We modified http_load so that it can compute the individual access latencies.

In addition, for the server-oriented scenario, 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 interval. The server implements the *Transcoding* policy as follows: if the average consumed bandwidth for the past half hour is more than the target bandwidth, the server serves proportionately lower quality variations of images until the consumed bandwidth exceeds twice the target bandwidth at which time the server denies further requests. 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. 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.

On the other hand, for the client-oriented scenario, we modified http_load to throttle the access rates to 1000, 3360 and 6720 bytes per second, which corresponds to 9600, 28800 and 56000 baud modems.

IV. RESULTS

A. Server-Oriented Scenario

First we analyze the ability of a web service to manage its bandwidth consumption using quality aware transcoding. Next, we analyze the ability of the bandwidth management schemes to dynamically provide differentiated web services to preferred clients.

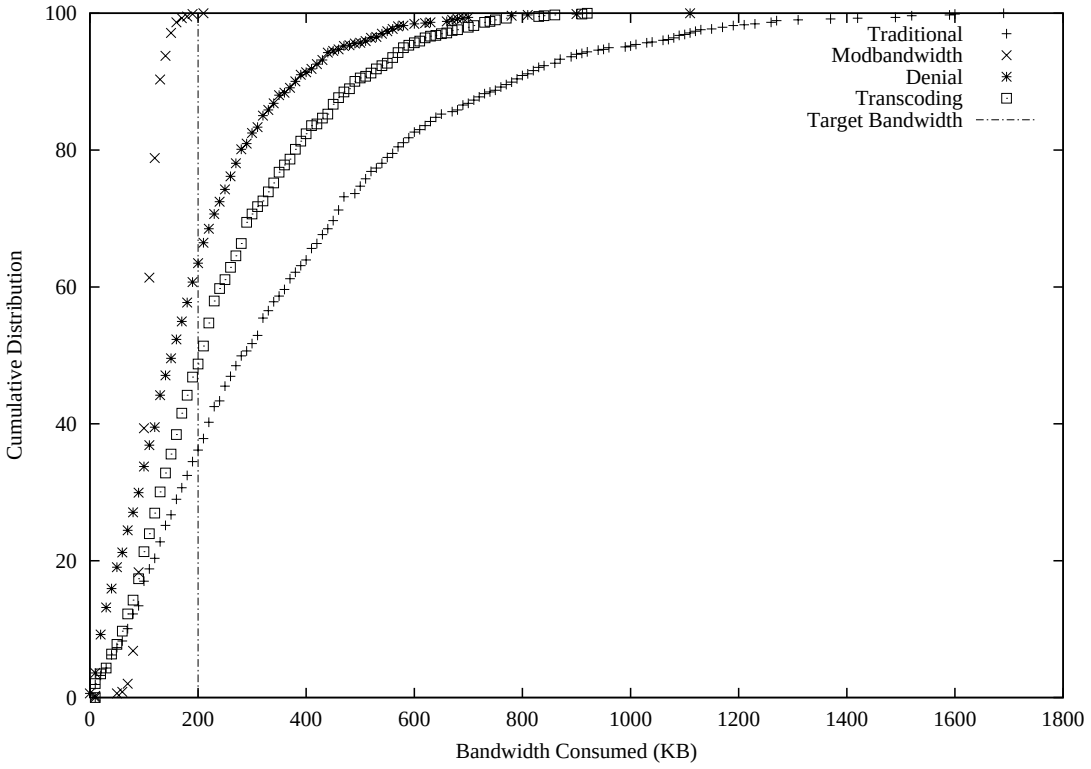
A.1 Bandwidth Control

In Fig. 5, we plot the bandwidth consumed by the server, for the various service policies described in Section III-D. Fig. 5(a) plots the cumulative distribution of the bandwidth consumed and Fig. 5(b) plots the bandwidth consumption with experiment time (smoothed using a bezier function).

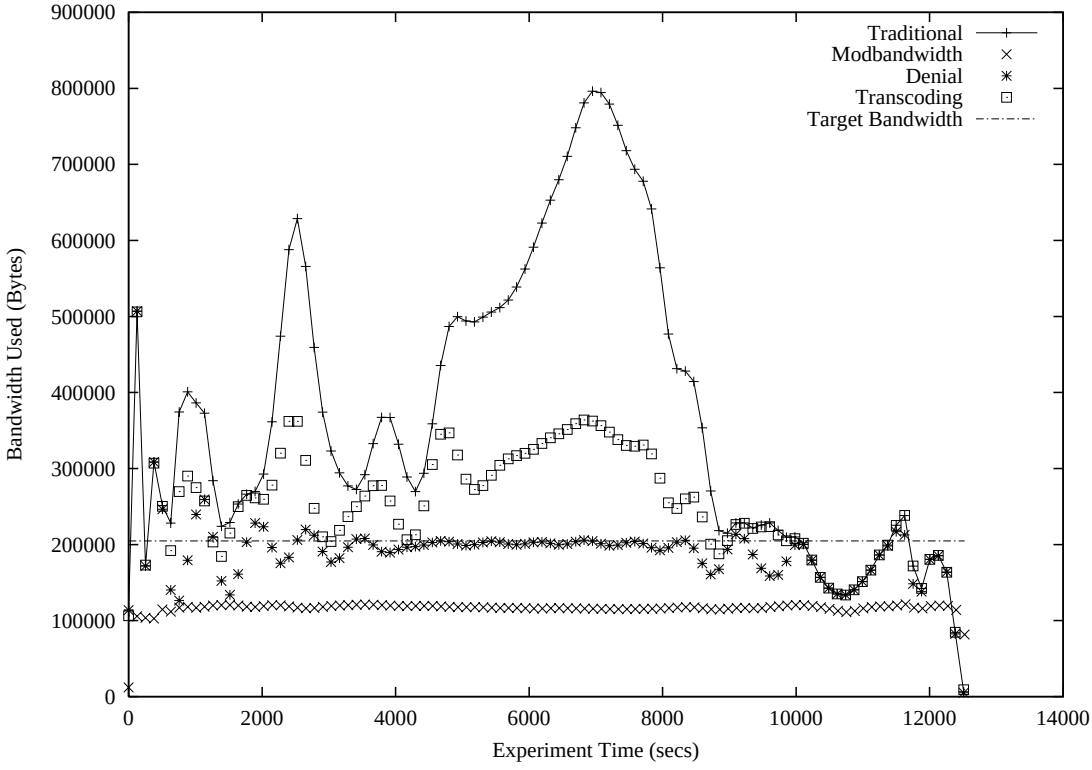
From 5(a), we note that *Traditional* overshoots the target bandwidth 65% of the time (dashed line at 200 KB), while *Modbandwidth*, *Denial* and *Transcoding* overshoot the target bandwidth 0%, 35% and 45% of the time, respectively. Our *Transcoding* bandwidth management algorithm reacts to average 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. We also note that *Modbandwidth* and *Denial* provide 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 use of network bandwidth, increasing client latency and consuming additional kernel resources (mbufs). Fig. 5(b), 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 orders of magnitude smaller than the requested bandwidth, (time=7000 seconds), the *Transcoding* scheme consumes as much as 400 KB/s because of the high difference between the requested bandwidth and target bandwidth. This limitation is largely a result of the smoothing function we employed to filter out short term bursts in client accesses.

Next, we measure the performance of the system using the Quality Factor of the images served to the clients and server service latencies. Figs. 6(a) and 6(b) plot the Quality Factor of the images and server service latencies as cumulative distributions, respectively. Denied requests are marked with a infinite service latency and Quality Factor of 0.

From 6(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 6(b), we note that the service latencies are quite high for the *Modbandwidth* scheme; 90% of the

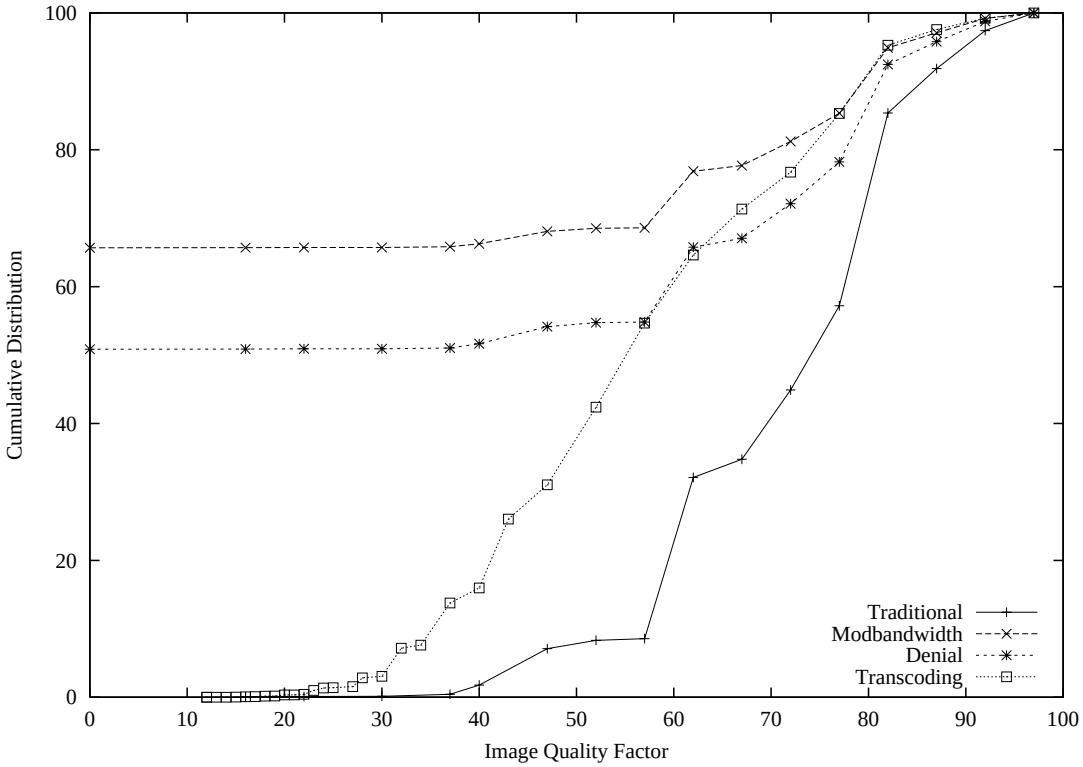


(a) Bandwidth Consumption Distribution

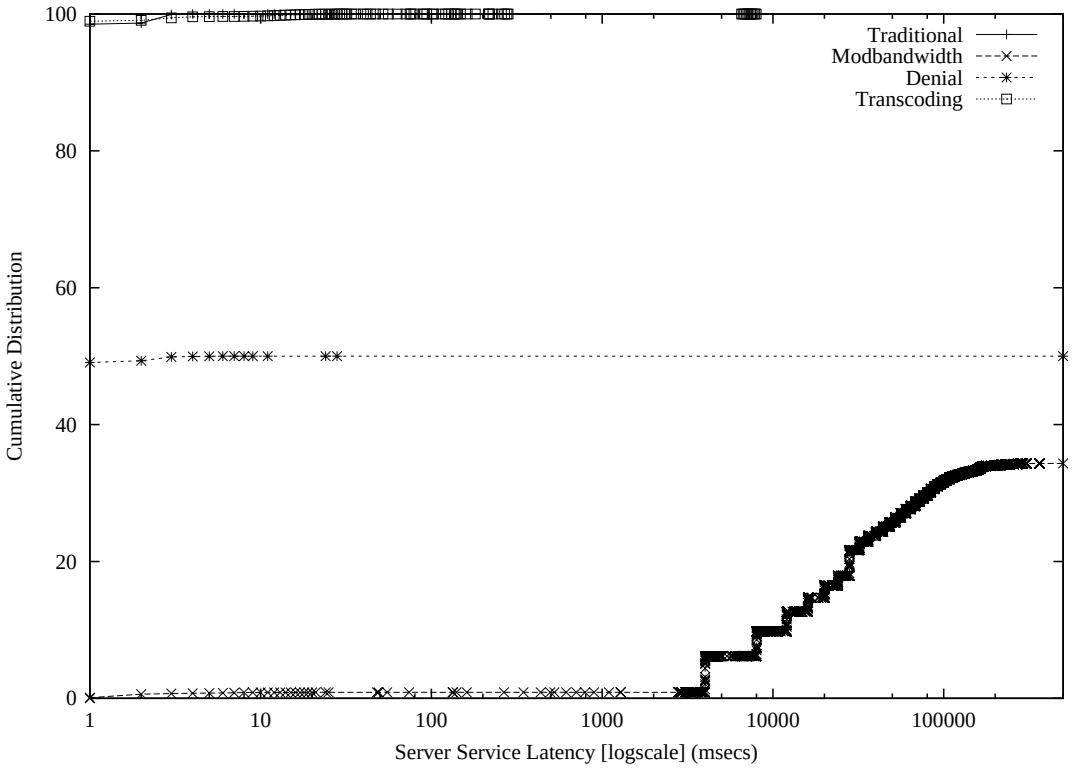


(b) Bandwidth consumed with time

Fig. 5. Server Oriented Scenario: Bandwidth Consumed (Target Bandwidth=200 KB/s)



(a) Image Quality Factor



(b) Client Access Latency

Fig. 6. Server Oriented Scenario: Bandwidth Control (Target Bandwidth=200 KB/s)

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.

A.2 Differentiated Web Service

Next, we analyze the ability of the bandwidth management 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 results are plotted in Fig. 7. Fig. 7(a), plots the bandwidth consumed by the server as a cumulative distribution while Fig. 7(b) plots the bandwidth consumed with experiment time. Fig. 7(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.

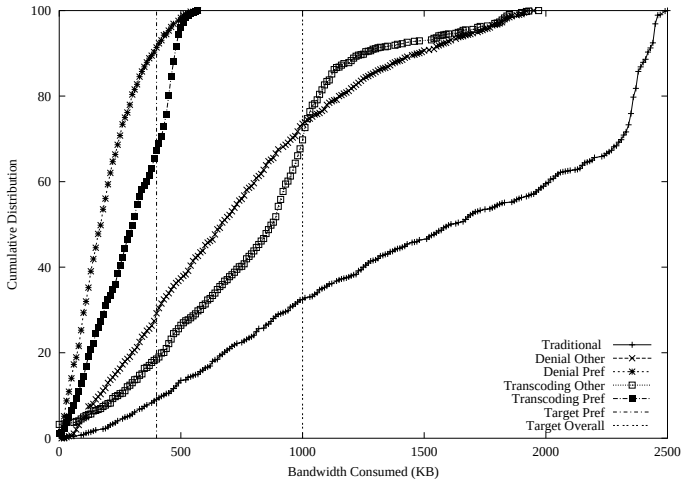
From Fig. 7(a), we note that the *Denial* and *Transcoding* provide bandwidth control for the different client classes. For *Denial* and *Transcoding* schemes, 90% and 65% of the preferred clients respectively, consume less than 400 KB/s in an epoch. Fig. 7(b), shows the results with experiment time for reference. Since the target bandwidth is closer to the maximum bandwidth, the system had better latitude in managing its bandwidth and hence better handles the heavy load (at time=4500 seconds). From Fig. 7(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.

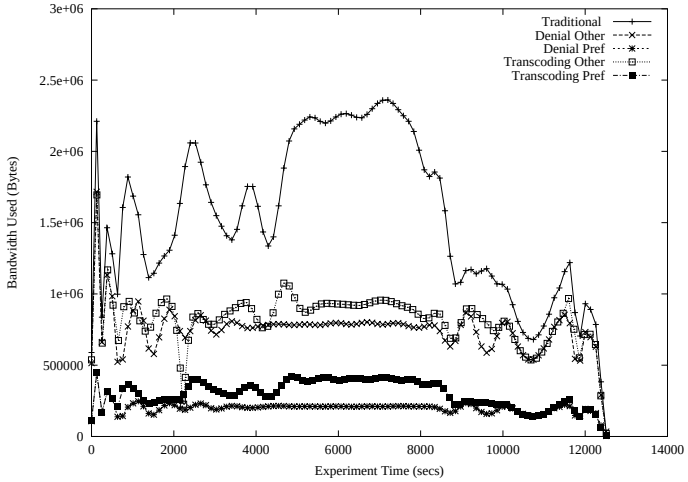
Hence we conclude that for a service offering differentiated QoS, *Transcoding* provides images of better Quality Factors to the preferred clients while degrading gracefully for non-preferred clients. *Denial* could not provide differentiated QoS without completely denying accesses to non-preferred clients.

B. Client-Oriented Scenario

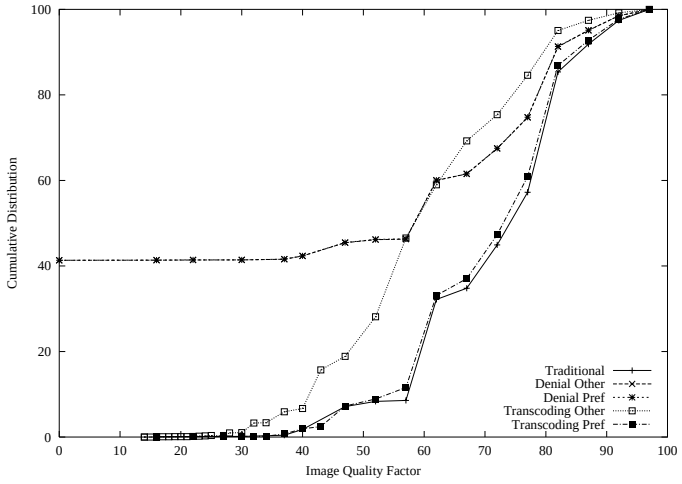
We measure the performance delivered to a client connected to its server using a slow link. The server used in the first experiment did not use any knowledge of the network used by the client in accessing the server. Hence the same image was served to clients,



(a) Bandwidth Consumed



(b) Bandwidth Consumed with time



(c) Image Quality Factor

Fig. 7. Server Oriented Scenario: Differentiated Service (Preferred Clients=20%, Reserved Bandwidth=400 KB/s, Target Bandwidth=1 MB/s)

regardless of the network used in accessing the images. Next we perform experiments to measure the server performance for the scenario where the transcoding server knows the current network characteristics to the client. We perform experiments using informed transcoding for images in our image collection. The knowledge of the network characteristics is used by the transcoding server to make an informed decision on whether transcoding is worth the effort.

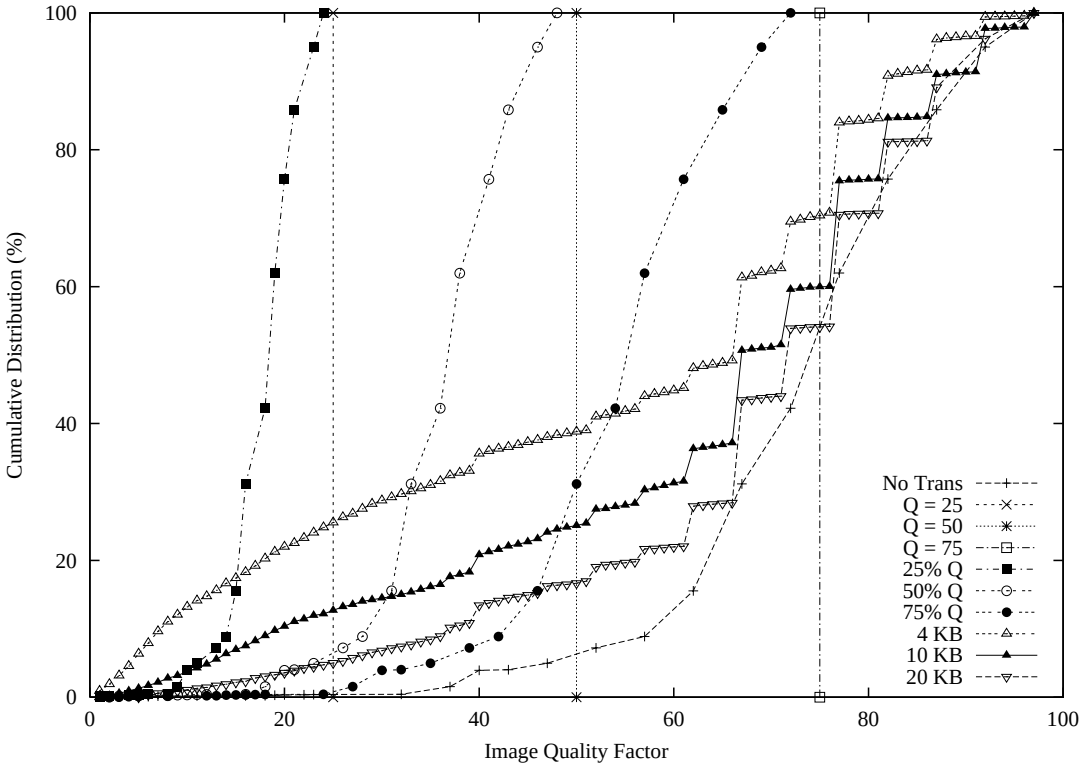
For the images in the NCSA and PhotoNet image collections, we plot cumulative distributions of the JPEG Quality Factors of the transcoded images and the number of bytes not transferred in Figs. 8(a) and 8(b) and Figs. 9(a) and 9(b) respectively. We want policies that can deliver images of Quality Factors similar to the Quality Factors provided by the non-transcoding server with savings in size. The results for the various policy alternatives are described below:

First we analyze the results for the *simple* policy applied to the images in the NCSA image collection. From Fig. 8(a), we note that *simple* policies transcode all images to a Quality Factor of 25, 50 and 75 respectively. From Fig. 4(a), we note that about 40% of the images have a JPEG Quality Factor less than 75 and about 7% of the images have a JPEG Quality Factor value less than 50. Hence, from Fig. 8(a), we infer that images whose initial JPEG Quality Factor values are lower than the transcoded Quality Factor are still transcoded to the higher JPEG Quality Factor value. The Quality Factors for these *simple* policies are represented by the vertical lines at 25, 50 and 75. These transcodings may produce an image that is larger than the original image, obviously without added information content. From 8(b), we verify that, for images transcoded to Quality Factor 75, about 8% of the images show an increase in image size of up to 10 KB (i.e. negative values for image size saved). For a policy that transcodes images to a Quality Factor 50, a smaller percentage of the images show an increase in image size of up to 7 KB. For a policy that transcodes to a Quality Factor of 25, 50 and 75, about 50%, 60% and 65% of the images had no savings in image size and 40%, 30% and 20% of the images in our collection saved from 0 through 10 KB respectively.

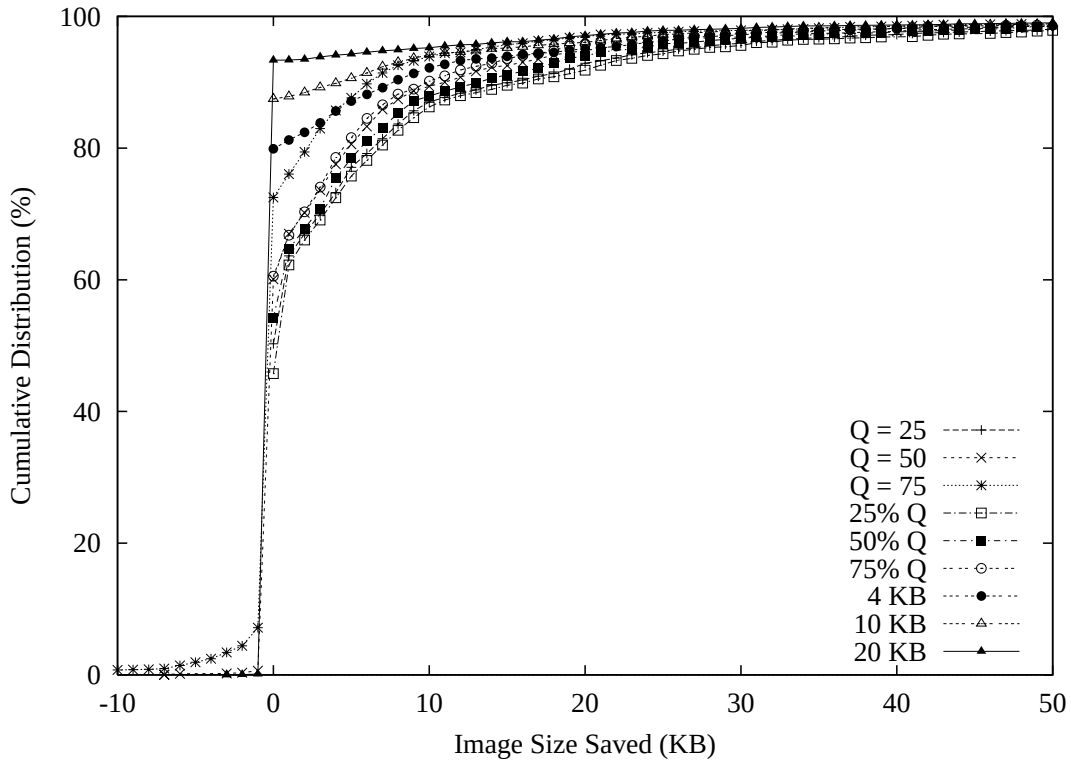
Fig. 4(a) shows that only about 5% of the images in the PhotoNet image collection have a JPEG Quality Factor less than 75. Similarly, Fig. 9(b), shows that transcoding images to Quality Factor 75, results in about 60% of the images increasing in image size of up to 30 KB. For a policy that transcodes images to a Quality Factor 50, a smaller percentage of the images show an increase in image size of up to 20 KB. Transcoding to a fixed Quality Factor of 25, 50 and 75 provided no savings in image size for about 10%, 10% and 40% of the images in our collection saved respectively.

The *relative* quality policies have access to the initial image JPEG Quality Factors and hence they transcode images to relative values of 25, 50 and 75 percent of the initial Quality Factors. These policies avoid transcoding images to a Quality Factor that is higher than the original Quality Factor.

For the images in the NCSA image collection, Fig. 8(a), shows that the JPEG Quality Factor of the transcoded images is worse than the corresponding *simple* policy. From Fig. 8(b), we note that for a transcoding that transcodes the images to Quality Factor

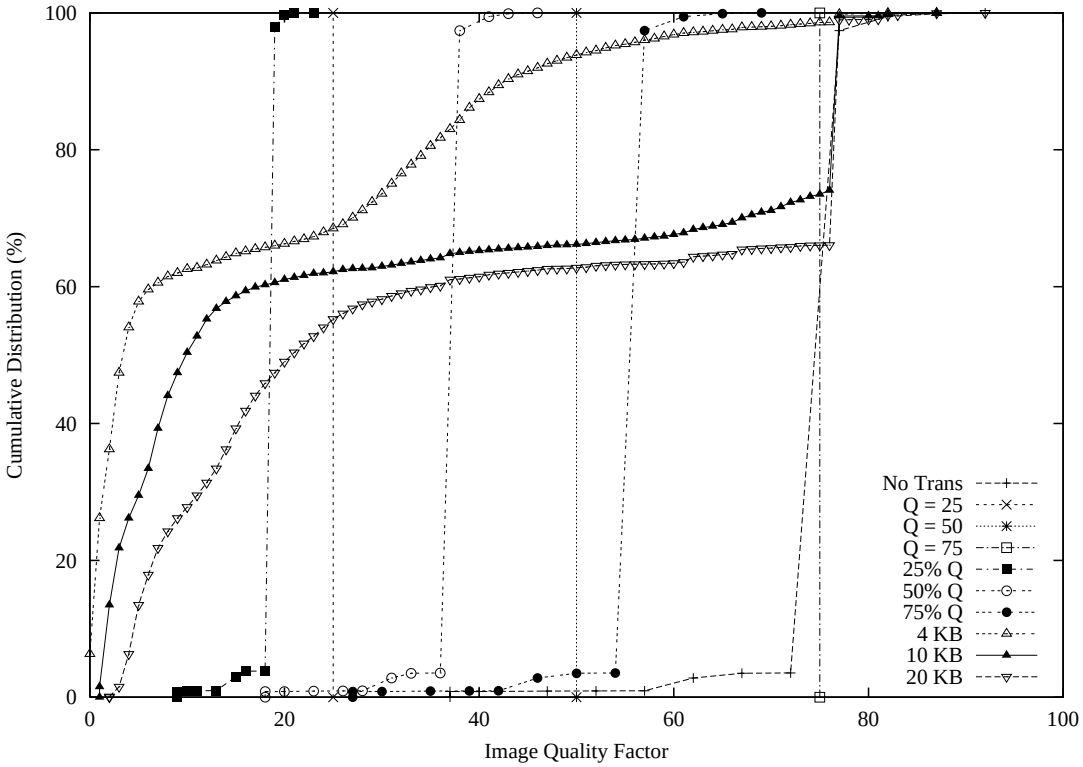


(a) Image Quality Distribution

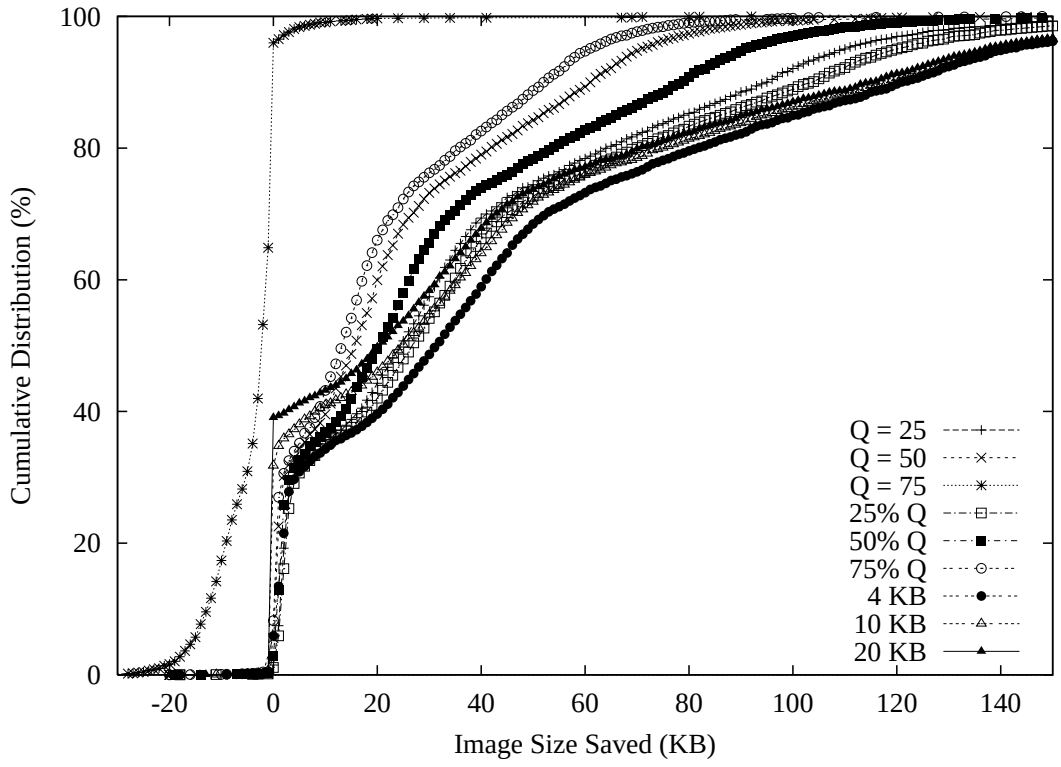


(b) Distribution of Image Size Saved (rounded down)

Fig. 8. Client Oriented Scenario: Perf. w/o knowledge of network characteristics (NCSA)



(a) Image Quality Distribution



(b) Distribution of Image Size Saved (rounded down)

Fig. 9. Client Oriented Scenario: Perf. w/o knowledge of network characteristics (PhotoNet)

values of 25%, 50% and 75% of the initial JPEG Quality Factor values, about 45%, 50% and 60% of the images had no savings in image size and about 40%, 30% and 30% of the images saved from 0 through 10 KB respectively. These values are similar to the results for the simple policy.

Next we consider the *relative* policies for the PhotoNet image collection. As with NCSA, Fig. 9(a), shows that the JPEG Quality Factor of the transcoded images is lower than the corresponding *simple* policy. However from Fig. 9(b), we see that more images show some decrease in size with these *relative* policies for transcodings to Quality Factor values of 25%, 50% and 75% of the initial JPEG Quality Factor values. Only about 5%, 5% and 10% of the images provided no savings in image size and about 40%, 30% and 30% of the images in our collection saved from 0 through 60 KB respectively.

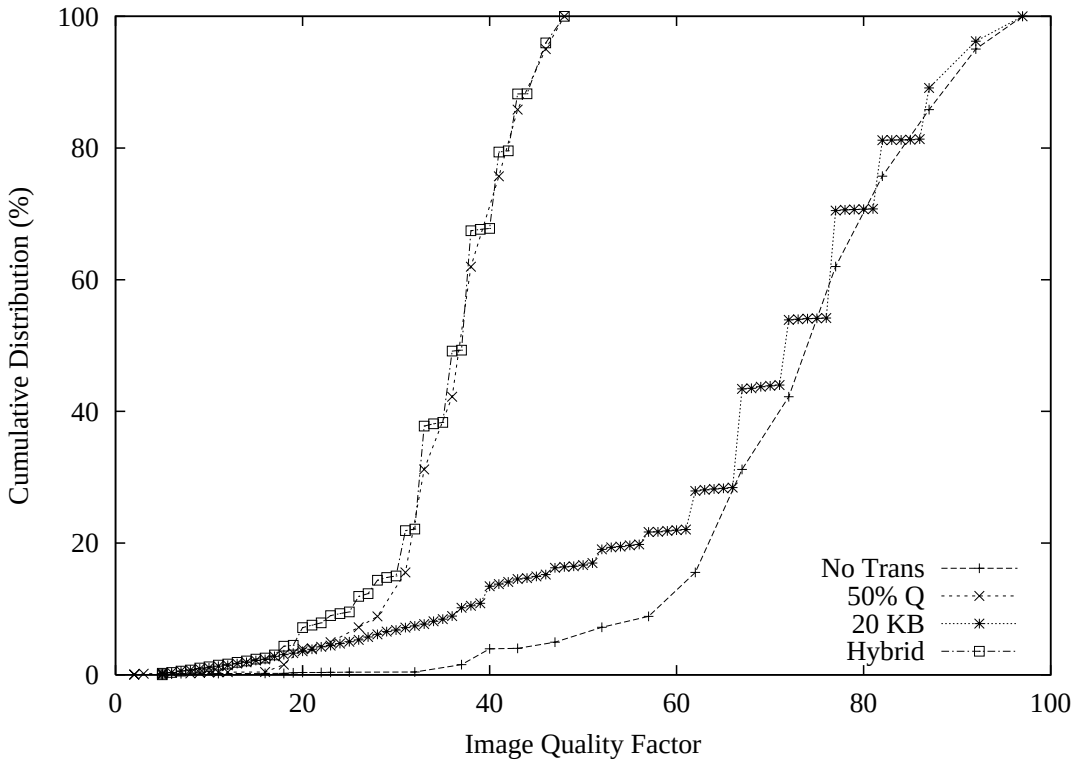
Next we analyze the results of transcoding to a *target size* for the images in the NCSA image collection. From Fig. 8(a), we note that the target size policies provide images with Quality Factors that closely match the Quality Factors of the images in the base, un-transcoded image collection. For a policy that transcodes images to a target size of 4 KB, only 12% of the images save 0 through 10 KB, while a policy to transcode to a target size of 20 KB provides no savings for 90% of the images with little savings in the range 0 through 10 KB. By avoiding unnecessary work for small images, this policy applies to the fewer large images.

Next we analyze the *target size* results for the images in the PhotoNet image collection. Again the resulting Quality Factors in Fig. 9(a) closely match the Quality Factors of the images in original image collection. Recall from Figure Fig. 4(b) that these images tended to be large. For a policy that transcodes images to a target size of 4 KB, 72% of the images save 0 through 60 KB, while a policy to transcode to a target size of 20 KB provides savings of 0 through 60 KB for 80% of the images. However, about 5% of the images save more than 140 KB.

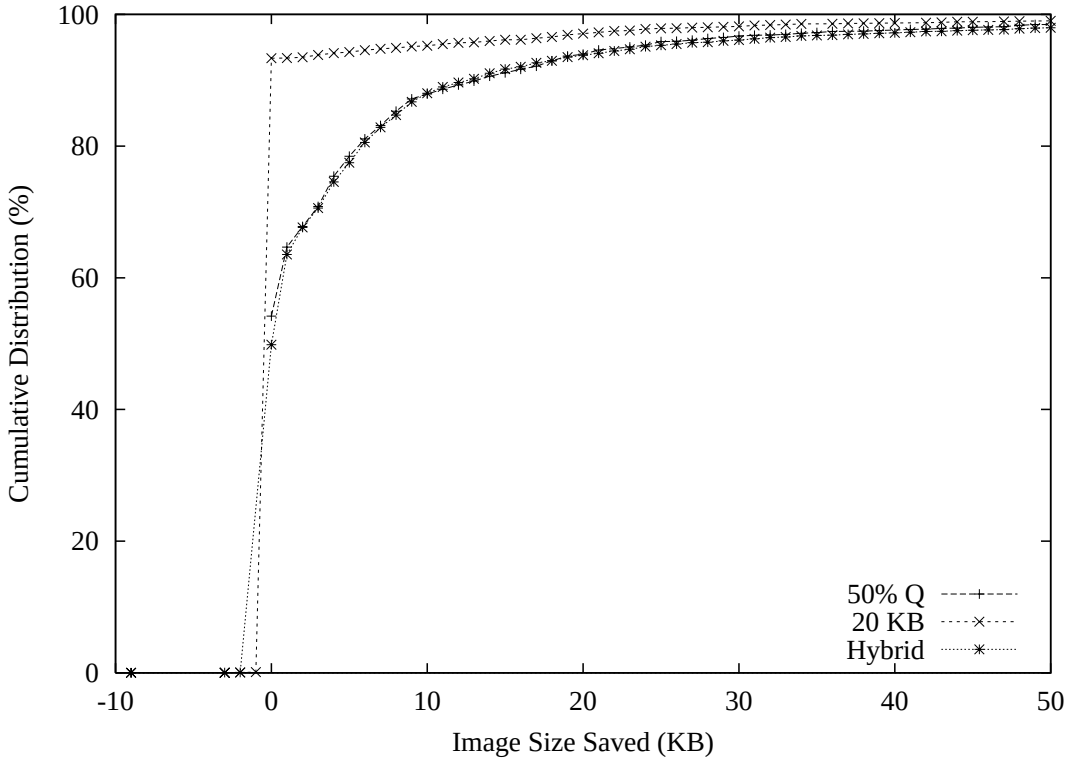
Hence we conclude that, for the images in both the NCSA and PhotoNet collections, transcodings that change the JPEG Quality Factor of an image to generate a target image size of 4 KB or 10 KB produce images that closely match the JPEG Quality Factors of the input images with savings in file size especially targetted to the larger files where it matters most. Transcodings that change the JPEG compression metric to be a fraction of the input image Quality Factor can save at least 0 through 10 KB for at most 40% of the images. Policies that transcode the images to ad hoc Quality Factor values can actually increase the transcoded image size.

B.1 Hybrid Policies

Based on the results of performing our experiments on the PhotoNet and NCSA image collections we conclude that a transcoding policy that converts images to a target size aggressively transcodes large images, generating big savings for large files, without reductions for already small files. On the other hand, a policy that uniformly transcoded all images to a fraction of the initial Quality Factor value produces savings for small and large files with greater loss in image information quality. These observations

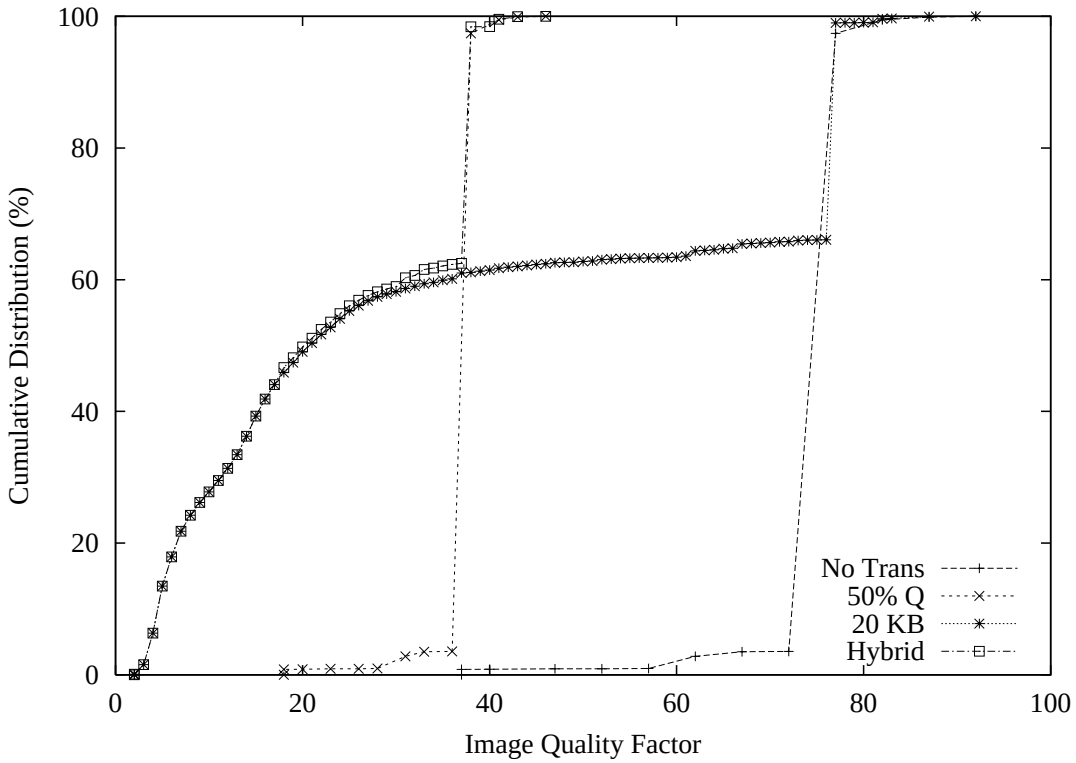


(a) Image Quality Distribution (NCSA)

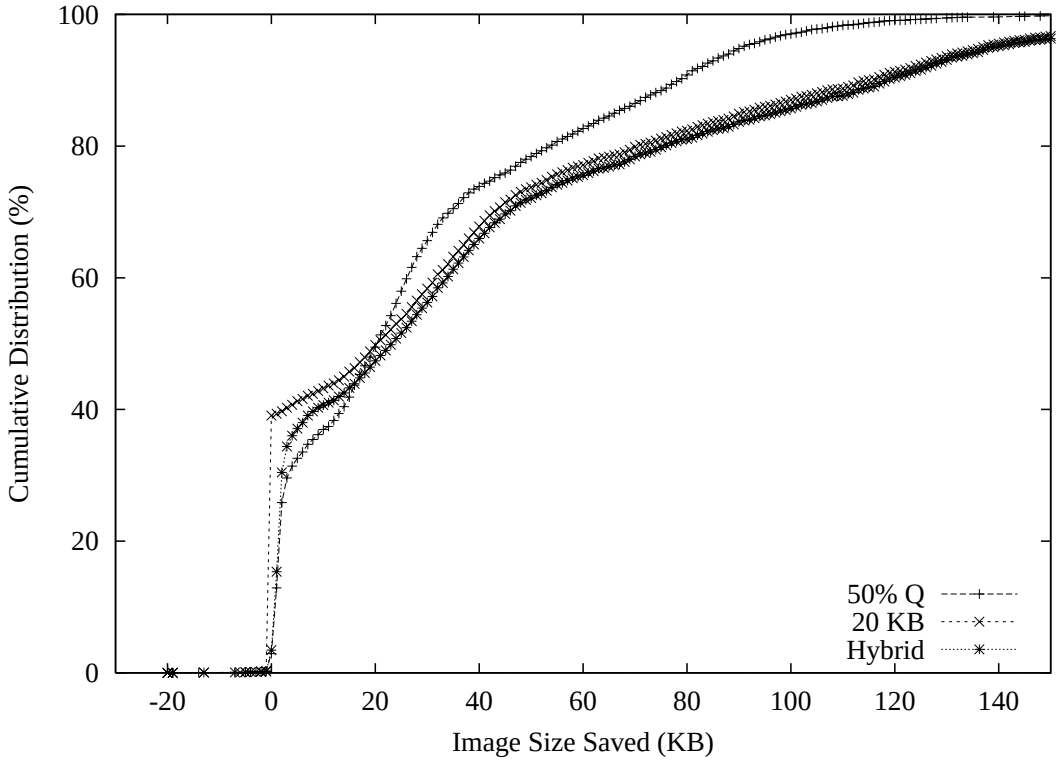


(b) Dist. of Image Size Saved (NCSA)

Fig. 10. Client Oriented Scenario: Perf. using a Hybrid Transcoding Policy (NCSA)



(a) Image Quality Distribution (PhotoNet)



(b) Dist. of Image Size Saved (PhotoNet)

Fig. 11. Client Oriented Scenario: Perf. using a Hybrid Transcoding Policy (PhotoNet)

make it possible for the servers to make a choice on how aggressively to transcode the various types of images. Servers can choose to aggressively transcode large images and reduce the Quality Factor for high quality art work or choose to aggressively transcode thumbnails. We explore one such hybrid policy to illustrate how such a choice is possible.

We performed experiments using a *hybrid* policy that transcoded images that are less than 40 KB using a transcoding policy that transcodes images to 50% of the initial JPEG Quality Factor. Images that were larger than 40 KB were aggressively transcoded to a target size of 20 KB. We plot the cumulative distributions of the information Quality Factor of the transcoded images as well as the savings achieved in Figs. 10 and 11 for NCSA and PhotoNet respectively. From these figures we note that the Hybrid policy provides information quality loss and savings similar to the policy that transcodes images to 50% of the original image Quality Factor values for the NCSA image collection. For images in the PhotoNet collection, this hybrid policy performs similar to a policy that transcodes images to a target size of 20 KB. These (expected) performance results demonstrate how policies can be composed to selectively apply to different classes of images.

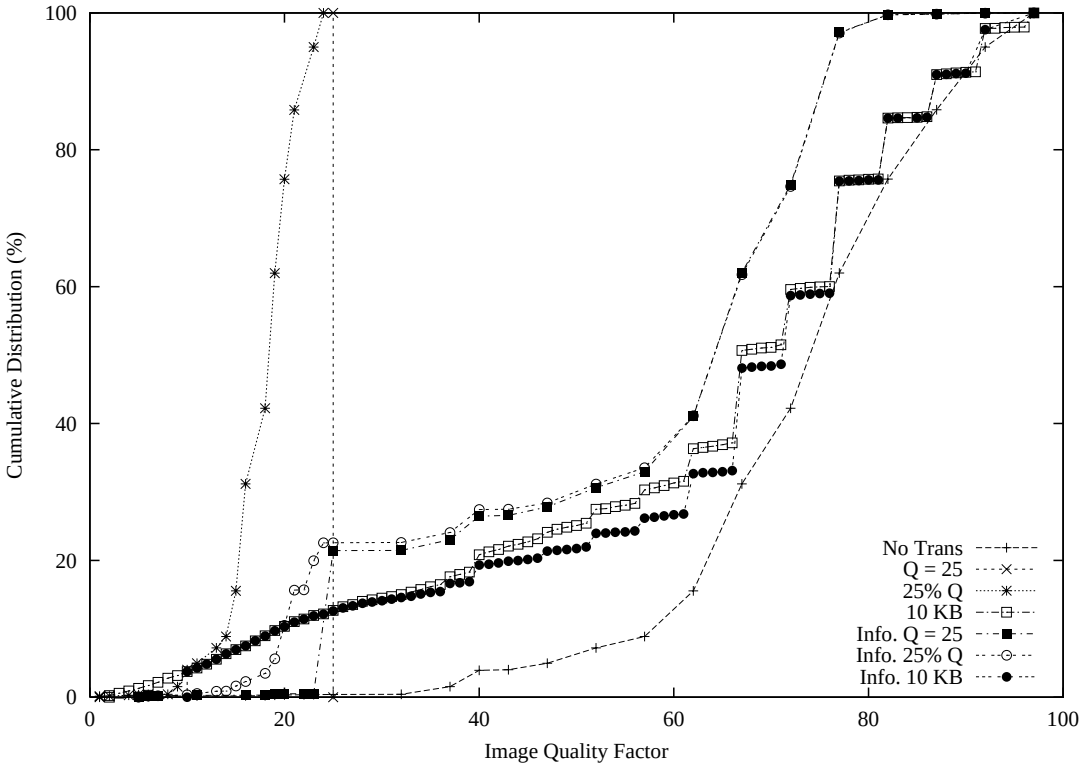
B.2 Server with knowledge of client link characteristics

A server that performs transcoding without taking the network conditions into account may perform unnecessary transcodings in the sense that the original image could be delivered with acceptable latency and cost. If clients access the proxies using a fast network, the proxies may transcode “small” images, even though sending the original image may be acceptable. The definition of a “small” image therefore depends on the current network environment and is defined by the size of images that can be served within a target end-to-end client latency. Unnecessary transcodings not only reduce the Quality Factor of an image, but can also place unnecessary CPU load on the server.

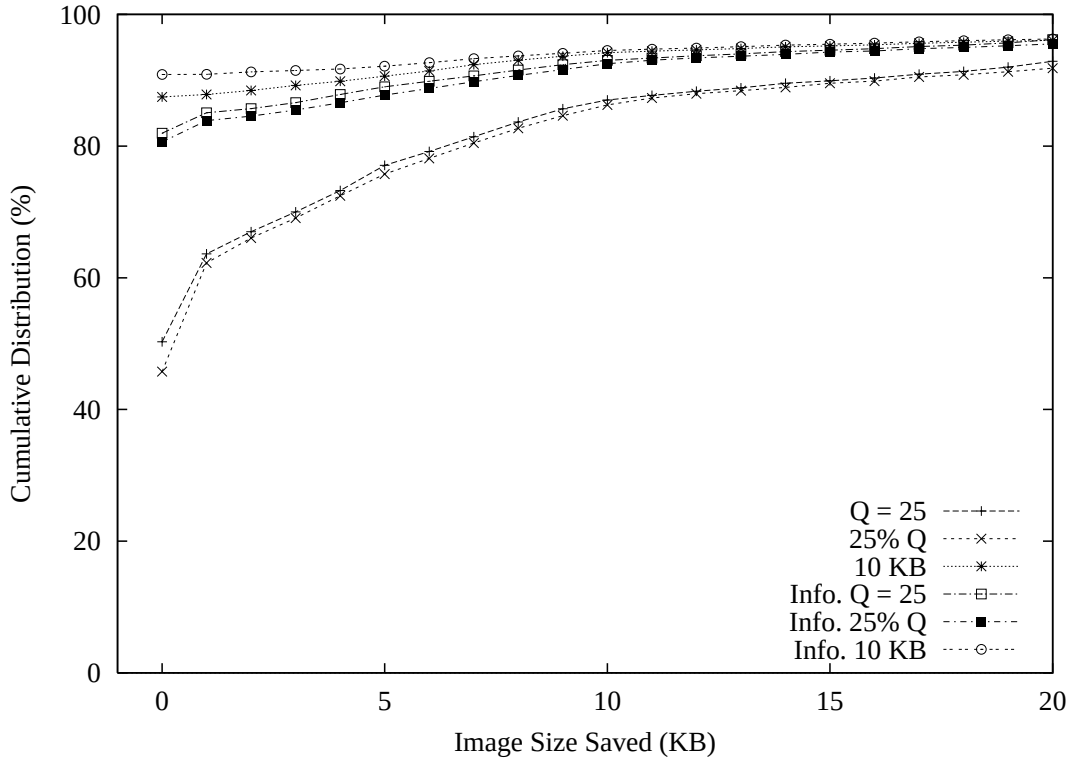
Our modified http_load client informs the server of the current network speed using our own HTTP header extension. If the server determines that a particular image can be delivered to the client within the acceptable latency of ten seconds, it will not transcode the image, even if the default policy perhaps indicates otherwise. The informed policy transcodes images that are deemed to be worth the effort. For our experiments, we use ten seconds as the target client latency.

We note that the informed policy transcodes large images even if it was predicted to not efficiently lose more in file size than information quality because any reduction in file size is deemed preferable for large images. Hence, we expect no change in the behavior of an informed transcoding for large images. We expect informed transcoding to avoid transcoding what qualifies for small files under a particular set of conditions.

We measure the performance of the system using the metrics of image Quality Factor and the image size saved. We subsequently investigate the server CPU load savings. To reduce the complexity of the graphs, we illustrate the effects of informed transcoding

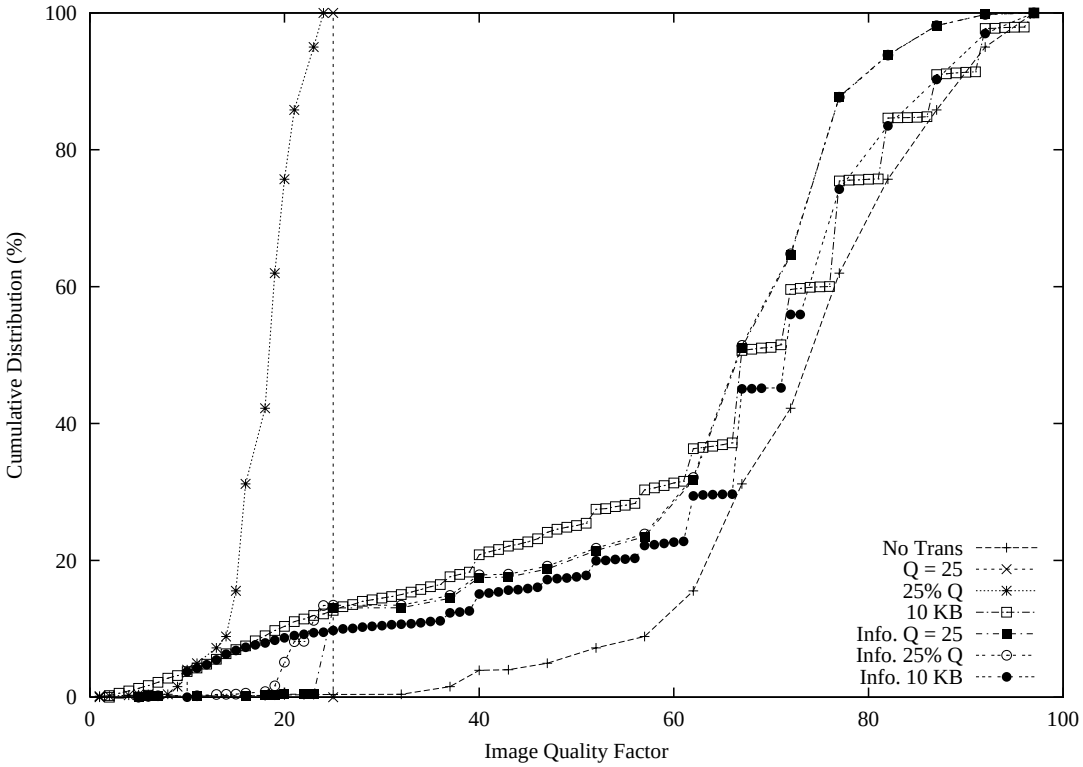


(a) Image Quality Distribution (9600)

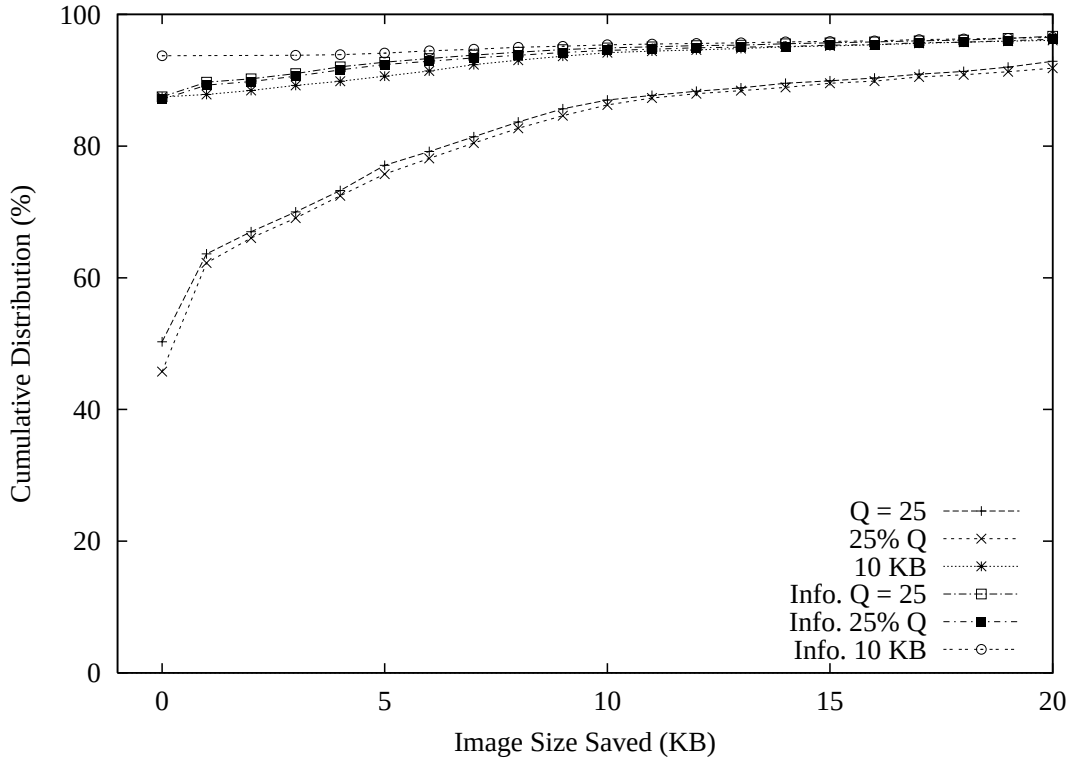


(b) Distribution of Image Size Saved (9600)

Fig. 12. Client Oriented Scenario: Server with knowledge of client link (NCSA; 9600)

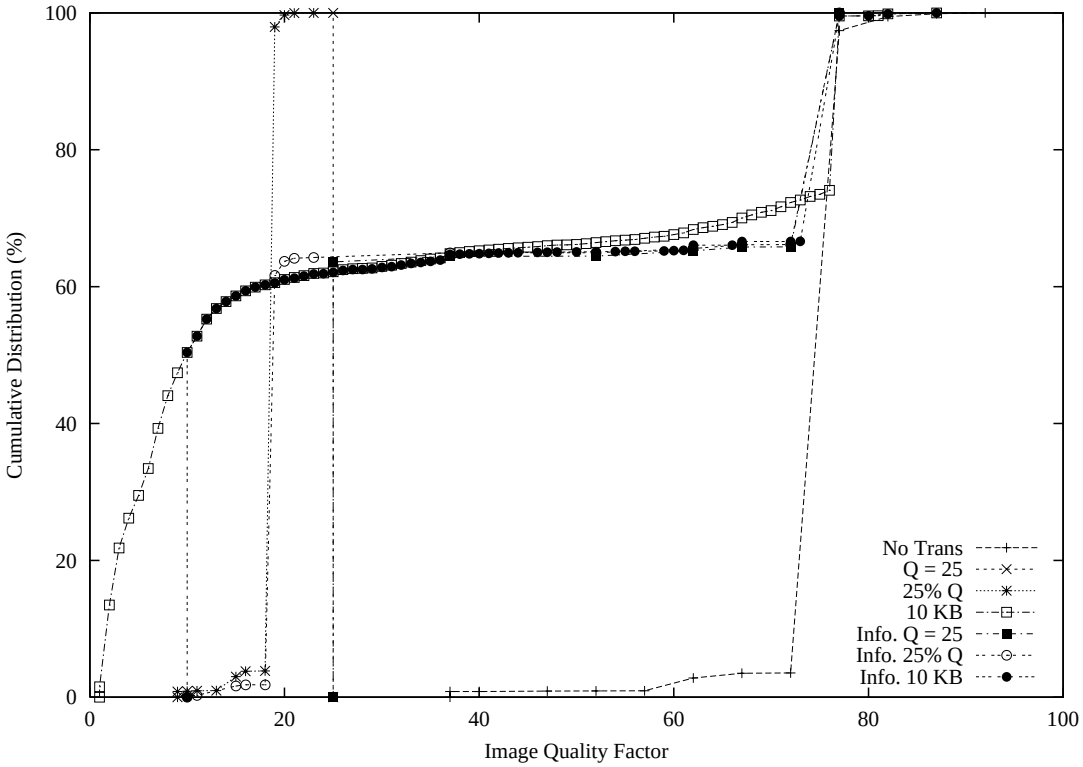


(a) Image Quality Distribution (28800)

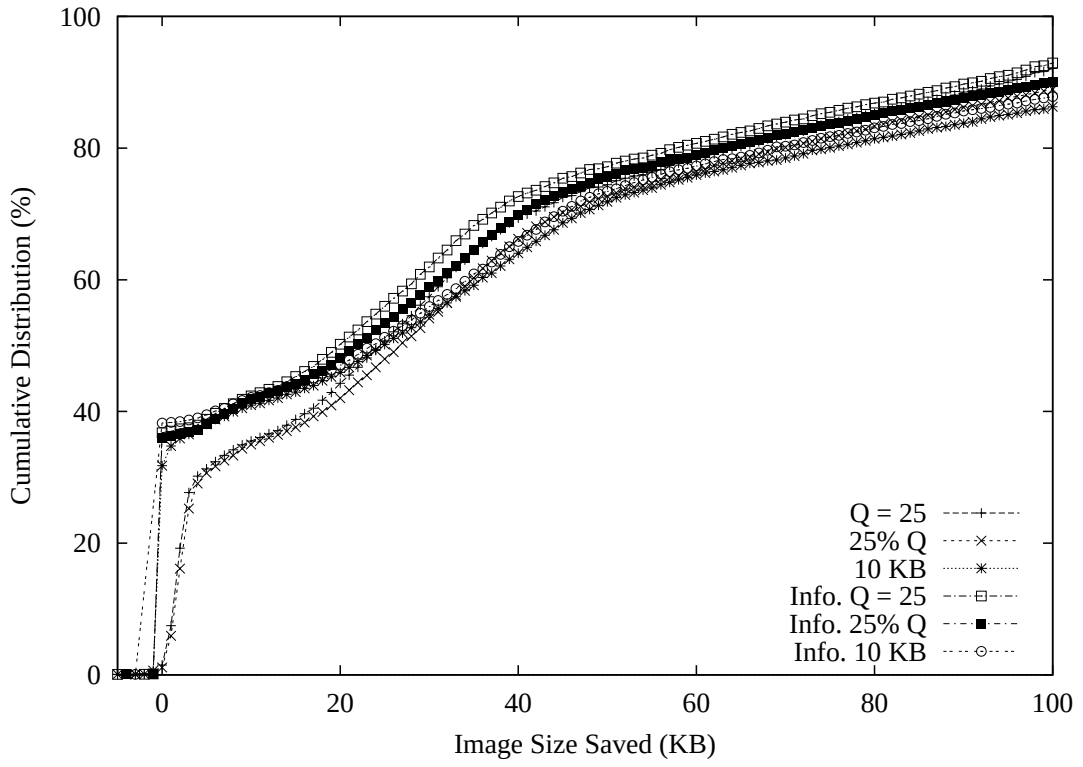


(b) Distribution of Image Size Saved (28800)

Fig. 13. Client Oriented Scenario: Server with knowledge of client link (NCSA; 28800)

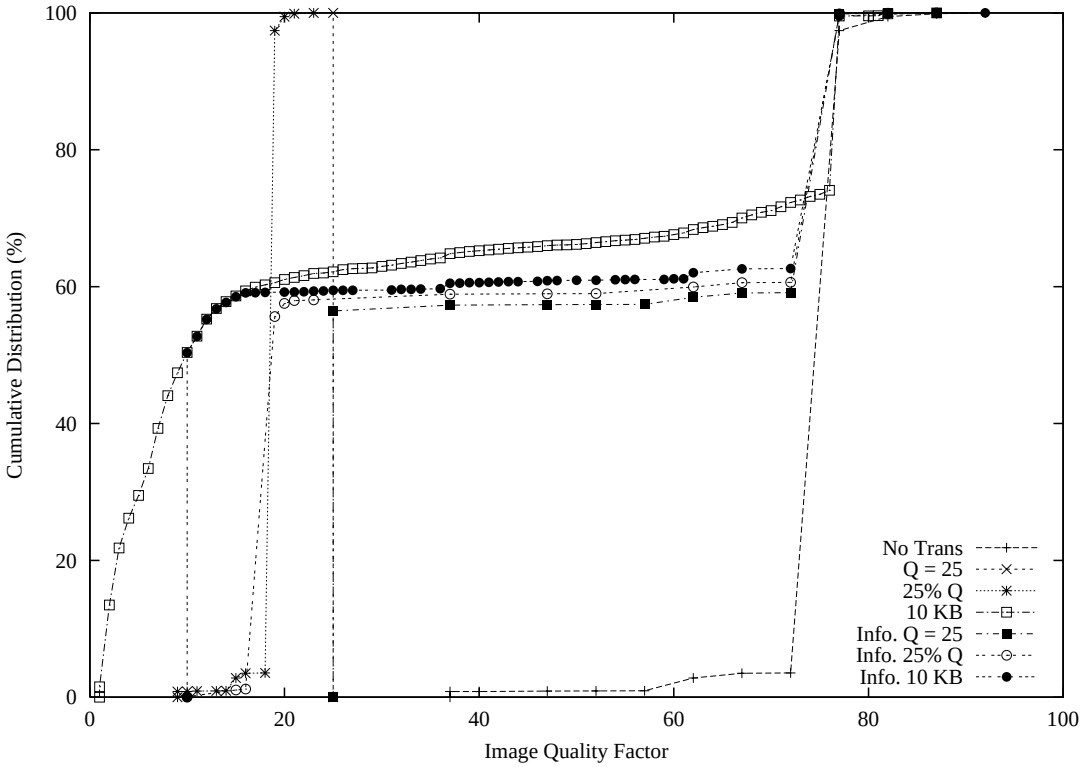


(a) Image Quality Distribution (9600)

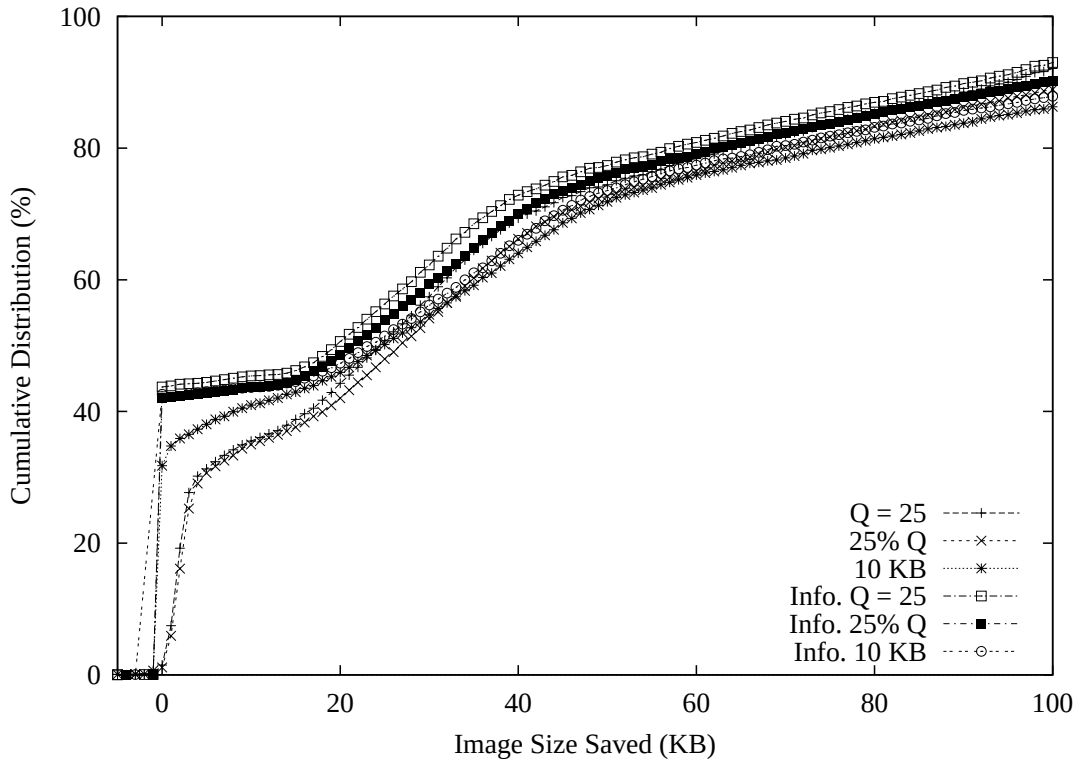


(b) Distribution of Image Size Saved (9600)

Fig. 14. Client Oriented Scenario: Server with knowledge of client link (PhotoNet; 9600)



(a) Image Quality Distribution (28800)



(b) Distribution of Image Size Saved (28800)

Fig. 15. Client Oriented Scenario: Server with knowledge of client link (PhotoNet; 28800)

for a policy that transcodes images to a fixed Quality Factor of 25, a policy that transcodes images to an image that is 25% of the Quality Factor of the original image and a policy that transcodes images to a target image size of 10 KB.

We perform experiments on the NCSA and PhotoNet image collection using the informed transcoding policy described in Section III-D. We plot the image size saved and the image Quality Factors as a cumulative distribution for clients accessing the server using a wireless and wireline modem (which corresponds to a 9600 baud and a 28800 baud modem) for the NCSA and PhotoNet image collection in Figs. 12 through 15.

First we analyze the results for the images in the NCSA image collection. From Figs. 12 and 13, we note that the informed policy treats small savings of less than 5 KB as unnecessary and hence informed transcoding policies that transcode images to 25% of the initial JPEG Quality Factor behave similar to a policy that transcodes images to a target size of 10 KB. A transcoding policy that transcodes images to 25% of the initial JPEG Quality Factor transcodes 60% of the images, while the informed counterpart only transcodes 20% and 10% of the images for clients accessing the server using a 9600 and 28800 baud modems respectively. A transcoding policy that transcodes images to a target size of 10 KB already does not transcode small files and hence informed policies do not make any difference for these images.

Next we analyze the results for the images in the PhotoNet image collection. From Figs. 14 and 15, we note that with informed policies, 35% and 42% of the images are not transcoded for clients accessing the server using a 9600 and 28800 baud modem. Since the PhotoNet images are predominantly large, informed transcoding does not make a significant difference except for the thumbnails.

Hence we conclude that for small and medium size images, informed transcoding can prevent unnecessary transcodings reducing computing load on the server as well as providing higher quality images to the client while maintaining a target response time. Informed transcoding uses the transcoding characterization and a knowledge of the characteristics of the link to the client and hence adds no significant overhead.

V. Conclusions

In this paper, we explore the use of informed transcoding to dynamically manage a web server's bandwidth consumption. We also explore an informed transcoding policy that utilizes knowledge of end-to-end network characteristics to deliver high quality multimedia content while maintaining predictable and user-settable access times.

We show that transcoding allows the server to manage its bandwidth without adding excessive latency or denying service. Transcoding allows the web server to provide differentiated service by allocating its bandwidth for different client classes. The service degrades the quality gracefully (and at different rates) for different client classes, while still managing overall consumed

bandwidth effectively.

We also show that transcoding allows a server to provide higher image information quality (as measured by Quality Factor in our experiments) to clients depending on the network used in accessing the server. We show that policies that transcode images to a fixed target size transcode large images aggressively and hence provide significant savings for the cases where it is most crucial. The transcoded images have JPEG Quality Factors that closely match the base case. Policies that transcode images to a percentage of the initial JPEG Quality Factors provide savings for small files with loss in information quality. Hybrid policies can use combinations of these policies to choose the level of saving preferred for a particular workload. Informed policies not only provide higher Quality Factor images to the client, but also reduce the load on the server by not performing unnecessary transcodings. We show that policies that aggressively transcode the larger images can produce images with Quality Factor values that closely follow the un-transcoded base case while still saving as much as 150 KB. A informed transcoding policy that has knowledge of the characteristics of the link to the client can avoid as many as 40% of (unnecessary) transcodings.

We are currently investigating techniques to allow web designers to specify the relative importance of various multimedia components of web pages, e.g., to allow the server to choose the transcoding level intelligently on a per-object basis.

REFERENCES

[1] 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.

[2] 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 ACM Multimedia 2000).

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

[4] 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.

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

[6] Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat, "Differentiated multimedia web services using quality aware transcoding," in *INFOCOM - Nineteenth Annual Joint Conference Of The IEEE Computer And Communications Societies*, Tel Aviv, Israel, March 2000, IEEE.

[7] Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat, "Multimedia Web Services for Mobile Clients Using Quality Aware Transcoding," in *The Second ACM International Workshop on Wireless Mobile Multimedia*, Seattle, August 1999, ACM SIGMOBILE.

[8] CompuServe Inc., 5000, Arlington Centre Blvd, Columbus, OH 43220, *Graphics Interchange Format (GIF) - A standard defining a mechanism forthe storage and transmission of raster-based graphics information*, June 1987.

[9] CompuServe Inc., 5000, Arlington Centre Blvd, Columbus, OH 43220, *Graphics Interchange Format - Version 89a*, 1989.

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

[11] Richard Han, Pravin Bhagwat, Richard LaMaire, Todd Mummert, Veronique Perret, and Jim Rubas, "Dynamic adaptation in an image transcoding proxy for mobile web browsing," *IEEE Personal Communications Magazine*, vol. 5, no. 6, pp. 8–17, December 1998.

[12] 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.

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

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.
[21] 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.
[22] 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.
[23] Yann Stettler, "Bandwidth management module for apache webserver," www.cohprog.com, Mar 1997.
[24] 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.
[25] 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.
[26] "WebQoS Version 2," www.hp.com/go/webqos.
[27] Nina Bhatti and Rich Friedrich, "Web server support for tiered services," in *IEEE Network*, Hui Zhang and Edward W. Knightly, Eds. IEEE Communications Society, September 1999.
[28] Jakob Nielsen, *Usability Engineering*, Academic Press, Boston, MA, 1993, (hardcover), 0-12-518406-9 (paperback).
[29] Tarek F. Abdelzaher and Nina Bhatti, "Web content adaptation to improve server overload behavior," in *Eigth International World Wide Web Conference*, Toronto, Canada, May 1999.
[30] Tarek F. Abdelzaher and Nina Bhatti, "Adaptive content delivery for web server qos," in *International Workshop on Quality of Service*, London, UK, June 1999.
[31] 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, Version of this paper was presented as the keynote talk in Infocom'99.
[32] Sandpiper Networks Inc., "Footprint adaptive content distribution service," www.sandpiper.net.
[33] Akamai Technologies Inc., "FreeFlow content distribution service," www.akamai.com.
[34] Armando Fox and Eric A. Brewer, "Reducing www latency and bandwidth requirements via real-time distillation," in *Proceedings of Fifth International World Wide Web Conference*, Paris, France, May 1996, pp. 1445–1456.
[35] 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.
[36] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, J. Eric Tilton, Jason Flinn, and Kevin R. Walker, "Application-aware adaptation for mobility," in *Proceedings of the 16th ACM Symposium on Operating Systems and Principles*, Saint-Malo, France, October 1997.
[37] Murray S. Mazer, Charlie Brooks, John LoVerso, Louis Theran, Fredrick Hirsch, Stavros Macrakis, Steve Shapiro, and Dennis Rockwell, "Distributed clients for enhanced usability, reliability, and adaptability in accessing the national information environment," Tech. Rep., The Open Group Research Institute, Cambridge MA 02142, 1998.
[38] Mika Liljeberg, Heikki Helin, Markku Kojo, and Kimmo Raatikainen, "Mowgli www software: Improved usability of www in mobile wan environments," in *IEEE Global Internet 1996*, London, England, November 1996, IEEE Communications Society.
[39] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K Raatikainen, "Optimizing world-wide web for weakly connected mobile workstations: An indirect approach," in *Proceedings of 2nd International Workshop on Services in Distributed and Networked Environments (SDNE'95)*, Whistler, Canada, June 1995.
[40] Jussi Kangasharju, Younggap Kwon, and Antonio Ortega, "Design and implementation of a soft caching proxy," in *3rd Intl. WWW Caching Workshop*, Manchester, England, June 1998.
[41] Rick Floyd, Barron Housel, and Carl Tait, "Mobile Web Access using eNetwork Web Express," *IEEE Personal Communications*, vol. 5, no. 5, October 1998.
[42] "Intel quickweb," www-us-east.intel.com/quickweb/.
[43] America Online Inc., "Johnson grace ART image format," .
[44] Spectrum Information Technologies Inc., "Fastlane," www.spectruminfo.com.
[45] Oracle Inc., "Oracle portal-to-go; any service to any device," Tech. Rep., Oracle Inc., www.oracle.com/mobile/portaltogo/, October 1999, Oracle business white paper.
[46] IBM, "Websphere transcoding publisher," www.ibm.com/software/webservers/transcoding/, 2000.
[47] B. D. Noble, L. Li, and A. Prakash, "The case for better throughput estimation," in *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, Rio Rico, AZ, March 1999.
[48] Tim Berners-Lee, R. T. Fielding, H. Frystyk Nielsen, J. Gettys, and J. Mogul, *Hypertext Transfer Protocol – HTTP/1.1*, January 1997.
[49] The National Laboratory for Applied Network Research, "A distributed testbed for national information provisioning," http://ircache.nlanr.net/.
[50] "Nielsen net rating service," http://nielsen-netratings.com/, March 1999.
[51] Robert Morris and Dong Lin, "Variance of aggregated web traffic," in *INFOCOM 2000*. IEEE, March 2000.
[52] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, November 1997.
[53] Jef Poskanzer, "Http load - multiprocessing http test client," www.acme.com/software/http_load/, 1998.
[54] "CNN Interactive: All Politics," cnn.com/ALLPOLITICS/, December 1998.
[55] Philip Greenspun, "photo.net," www.photo.net, December 1998.
[56] "Star Wars - Episode I," www.starwars.com/episode-i/, December 1998.