Multimedia Web Services for Mobile Clients 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

Mobile applications will increasingly depend upon multimedia information originating in the web and attempt to access this data over wireless networks that are more expensive and slower than typical wireline internet access from desktops. Transcoding is an important technique that can allow network proxy servers to offer differentiated service by customizing the delivered object size for the network bandwidth available on the "last hop" to the mobile client. We exploit technology that we had previously developed that characterized the quality versus size tradeoffs in transcoding JPEG images. This technology supplies more information for use in transcoding policy decision making. We evaluate the performance benefits of incrementally incorporating this information in a series of transcoding policies.

The principal contribution of this work is the demonstration that it is possible to use informed transcoding techniques to balance the need for good quality of multimedia content while reducing consumed network bandwidth and server CPU overhead. 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.

1 Introduction

An explosion in the number and variety of mobile devices is dramatically changing the world of computing. Mobile computing devices range in complexity from laptops to palmtops to PDAs to smart-phones to fully functional jewelry with computational resources.

Many applications that will run on these mobile devices will need or at least benefit from sources of information that originate outside of the device. Many mobile applications use the mobile device as a window into vast amounts of data that can be delivered via the internet, particularly in the form of Web content.

Mobile clients primarily access the internet using wireless technologies such as cellular (analog), CDPD, CDMA, GSM, Ardis, and Ricochet networks. The wireless network technologies have constraints on network bandwidth, connection reliability, network latency and access costs. Accesses from cellular modems using technologies such as GSM typically operate in the range up to 19.2 Kbps with speeds projected in the near future averaging 28.8 Kbps [27]. Mobile networks are expensive. For example, CDPD offered by our local service provider (GTE) costs between 6 and 12 cents per Kbyte depending on the level of usage.

The problem of slow and expensive networks is compounded by the large size of multimedia objects that are becoming such a prevalent part of web content. Studies [1] have shown that the average web page is about 61 Kbytes in size. Accessing 61 Kbytes using CDPD would take about 8 minutes and cost around 5 dollars. For many users, such access costs are prohibitive.

In general, there is a huge mismatch between the rich multimedia content available on the World Wide Web and the characteristics of wireless technologies that are used to access the Web; mobile users would like to keep the access wait and cost from becoming unbearable. Therefore, it is important to consider how to access WWW information while retaining the ability to go mobile, connected by expensive and slow wireless networks.

In such a scenario, differentiated service can allow the system to provide a more appropriate level of service for a mobile user based on the current network environment. Differentiated service means that proxies can match object sizes with the network bandwidth available on the last hop to mobile clients. In this way, a network proxy server can provide different versions of the same object to different clients. For example, the proxy server can choose variations of an object such that the objects are served at a uniform latency of less than 5 seconds based on the type of network link used in accessing the web.

The feasibility of such a differentiated service scheme depends on the ability to produce a range of variations for any object so that the server can choose the correct variation for the current network operating environment. Even though the content provider can manually provide a number of different variations for use by the system, an automatic technique may be preferable to allow the system to dynamically adapt to variability in client characteristics and network performance.

One promising technique for providing differentiated quality of service is transcoding which can be used to serve variations of the same multimedia object at different sizes. Transcoding is defined as 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 be useful in providing differentiated service, we need to understand the tradeoff characteristics: the information quality loss, the computational overhead required in computing the transcoding and the potential space benefits. To illustrate for one

specific case, we previously characterized the information quality tradeoffs, the computational requirements and the potential space gain of a transcoding that changes the JPEG [22] compression metric [3]. It has been shown that the JPEG Quality factor parameter reflects a user's perception of image quality [6, 13]. Without such characterization, transcoding policies do not have the ability to measure information quality of an image. Hence, current systems transcode images to ad hoc Quality factor values, potentially leading to an increase in the image size for certain images. Current systems have countered this 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 the image to a fraction of the original image Quality factor or transcode images to a target size. We also characterized the computational costs and the space benefits possible with a transcoding. Such a characterization allows us to explore an informed policy that uses information about the link characteristics to a client to dynamically decide if transcoding will be worth the effort for a particular request. Such characterization enables the system to make an informed decision of serving the right object for the end user.

In this paper, we investigate the potential benefits and overhead in providing differentiated service for the web, using a proxy server that performs quality-aware, informed transcoding for mobile end users accessing the web via wireless networks. 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.

For our experiments, we modify the Apache web server [2], 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 JPEG compression metric as the informed transcoding technique. We assert that the 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 balance the demand for quality of multimedia content while reducing consumed network bandwidth and server CPU overhead. We show that policies that transcode images to a target file size produce images with Quality factor values that closely follow the untranscoded base case by not transcoding small images while still saving as much as 150 KB by aggressively transcoding large images. On the other hand, transcoding policies that transcode images to a percentage of the original Quality factor values offer savings in image size of up to 10 KB for 50% of the images files at the expense of the maximum image Quality factor values and savings for large files. We find that, transcoding to ad hoc Quality factor values, as is the current practice today, can actually transcode images to a larger image size because of the lack of information about the initial image Quality factor. A transcoding policy that has knowledge of the characteristics of the link to the client can avoid as many as 40% of (unnecessary) transcodings. We also show that transcoding operations are computationally feasible in a proxy server.

Our results make it possible to make choices on how aggressively a proxy transcodes images of different sizes. Based on the characteristics of static policies, we describe a hybrid policy that transcodes images that are less than 40 KB to a fraction of the image Quality factor and larger files more aggressively. We show that such hybrid policies can help balance the client constraints for images available on the internet.

The rest of the paper is organized as follows: Section 2 reviews our previous work as the necessary background and places our work in context to other related work. Section 3 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 4 with conclusions and future work in Section 5.

2 Related Work

2.1 Quality Aware Transcoding

Quality aware transcoding is the enabling technology for our research. Earlier systems have used a number of transcoding techniques depending on the intended usage. Transcoding operations are often performed to fit an object to the characteristics of the display device. Images have been transcoded to thumbnails, grayscale, progressive formats as well as transcoded to textual information. Our focus is on transcoding to reduce bandwidth requirements on the wireless link. Very little work has been done in quantifying the actual information loss and computational characteristics of the transcoding operations. In our earlier work [3], we characterized the tradeoffs inherent to a transcoding that changes a JPEG compression metric, such as the JPEG Quality factor.

Reconstructing the original Quality factor that was 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) [15] 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 [6, 13].

Next, we characterize 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 does 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 efficiently at a significantly better percentage than the base case of all the images.

For our experiments, we used realistic image collections from a variety of typical web sites to validate our results.

Such characterization is central to applications that wish to use informed transcoding. Indeed, informed transcoding forms the basis for our ability to provide differential quality of service.

2.2 Transcoding Network Proxies

A number of systems have used transcoding to fit images to the current operating environment. None of the systems used an informed transcoding and hence performed ad hoc transcoding without an explicit understanding of the tradeoffs and potential gains.

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.

Odyssey [20] manipulated the JPEG Compression metric as a distillation technique for a web browser that adapts to changing network environments. Without the ability to measure the initial Quality factor of an image, they assumed that the original image is of JPEG Quality factor 100 (Fidelity = 1.0) and conclude that, at low bandwidth, JPEG Quality factor 50 is the best possible fidelity. However, those results are dependent on the initial JPEG Quality factor and without the ability to measure the initial JPEG Quality factor, the results are only valid for images that have high initial quality values. As we note later on, this transcoding policy can create images that are larger than the original un-transcoded image depending on the Quality factor of the image accessed.

Noble et al. [19] describe a system wherein the bottleneck bandwidth need not be within the last hop to the client. They describe a method that can provide a better estimation of the throughput so that the bottleneck bandwidth can be accurately determined. Their results complement our work by helping the proxies make better informed decisions. For our experiments, we have assumed that the last hop to the client is the network bottleneck. Their end-to-end throughput information can be used along with our informed transcoding to provide better end-to-end decisions.

The GloMop [7, 8] project used transcoding to generate thumbnails on the fly to speed up image access from slow modems. The Soft Caching project [14, 21] used progressive JPEG to produce lower quality images at the same spatial resolution. Caubweb [18] described a generic proxy service with content type specific application transducers that perform filtering operations such as transcoding. The Mowgli project [17, 16] described content type specific lossless and lossy compression techniques to improve the web experience over wireless mobile links. However, those systems do not use an informed transcoding technique that can quantify the information loss from the chosen transcoding operation.

Commercial products such as WebExpress [5] from IBM, QuickWeb technology [12] from Intel and Fastlane [24] from Spectrum Information technology have used various forms of compression and transcoding operations to improve web access from slow networks.

3 Experiment Objectives and Design

3.1 Objectives

We designed our experiments to answer the following questions. First, what is the delivered performance, as defined by image quality and bandwidth savings, to end users of a slow (wireless) connection? We consider the following scenarios: i) a base case with no transcoding, ii) a simple transcoding proxy where no method is used to compute image information quality values and no quantification of the transcoding trade-off is done (e.g., only transcode to ad hoc values such as JPEG Quality factor values of 25, 50 and 75), and iii) a relative transcoding policy where we exploit the ability to measure the information quality factor of an image. (e.g., transcode to a fixed image size or a percentage of the initial image size/quality factor).

Next, what is the performance delivered to the end user of a slow (wireless) connection for a proxy aware of available network bandwidth so that it can perform *informed transcoding*? Transcoding characterization described in [3] allows us to make decisions on the utility of transcoding for a particular operating environment. With computational cost characterization, we transcode only if we find that it is appropriate for an estimate of the current network bandwidth. With efficiency characterization, we transcode only if the image is likely to lose more in size than in quality. The final, related question we consider is the computational overhead of such

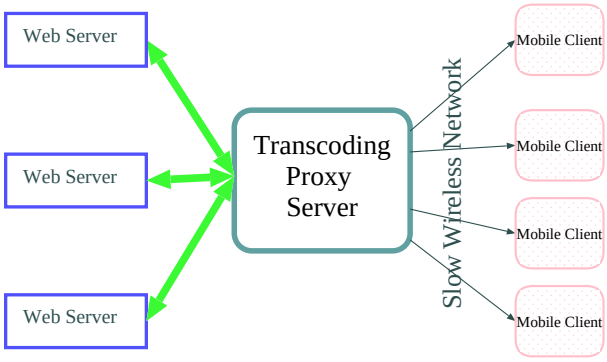


Figure 1: System Architecture

decision making on a proxy server performing informed transcoding? That is, can proxies keep up with dynamic transcoding. We measure proxy throughput to analyze this question.

3.2 System Architecture

The system that we envision is described in Figure 1. The important components of this system are:

Web Server The web servers hold the information that is consumed by the mobile clients. The web server is connected to the transcoding proxy servers using a fast network link.

Transcoding Proxy Server Much of the web content is dynamically generated (maps, stock charts etc) or un-cacheable (sites selling access to multimedia contents such as images or movies). For our purposes, the transcoding proxy is transcoding un-cacheable images for consumption by clients using slow, expensive and unreliable networks. Since the images are un-cacheable, the proxy transcodes all images on the fly.

Slow Network Clients access the network using slow, expensive and unreliable networks. Cellular technologies [27] typically provide access speeds of 9600 bps, while home modem users typically connect using a 28.8K or 56K modem.

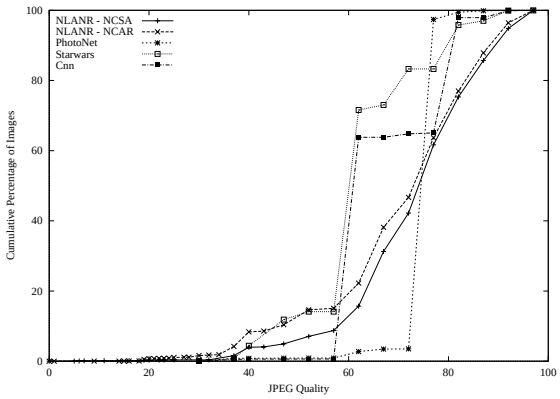
3.3 Performance Measures

The type of differentiated service our proxy 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 net bandwidth available to the clients. In such an operating environment, the following measures capture the performance of a network proxy server.

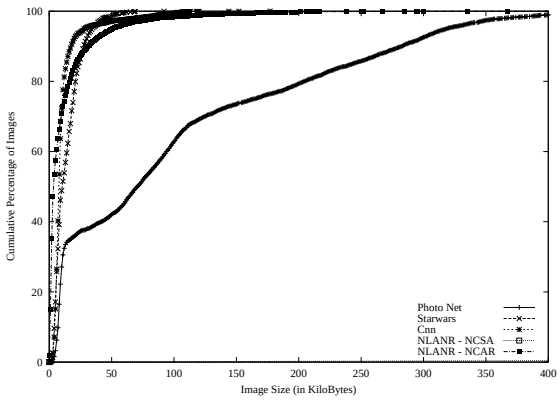
1. Image information quality
   Since transcoding trades off image information quality for size, the Quality factor of images served gives an indication of the quality tradeoff. The goal is to maintain as much information quality as possible, within the constraints of the available network bandwidth.

2. Number of bytes not sent
   This measure 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.

3. Server Throughput
   Measures such as bytes transferred per second and objects



(a) Cumulative distribution of image JPEG Quality factor



(b) Cumulative distribution of image File Size Distribution

Figure 2: Workload Characteristics

served per second capture the throughput of the proxy server. The throughput of the proxy server gives an indication of the cost and feasibility of deploying transcoding as a general multimedia access mechanism for mobile clients.

3.4 Experimental Setup

3.4.1 Workload

For our experiments, we use the Squid proxy logs from the NLANR [26] proxies. NLANR proxy caches make their traces (with actual, unscrambled URL's) publicly available for the past seven days. For our study, we utilize two sets of traces that were collected on Mar 23, 1999:

1. NCAR at Boulder, Colorado serves the US domains ending in .edu, .gov, .mil, .org and .us. For our experiments, we used 13711 JPEG images totaling 205 Mbytes, downloaded from this cache access log.

2. NCSA at Urbana-Champaign, Illinois serves the .com US domain. For our experiments, we used 11771 JPEG images totaling 171 Mbytes, downloaded from this cache access log.

We also consider synthetic access patterns consisting of requests for images available from typical web sites:

1. News Site: These sites encounter heavy traffic, especially when significant news items break. These news sites would rather not serve images if there is a choice between serving images and textual information. When significant stories break, sites such as MSNBC and CNN have used fewer graphics to conserve bandwidth. In general, we expect the images to be small and of low quality. For our experiments, we used 6217 JPEG images totaling 70 Mbytes, downloaded from Cnn.com [4].

2. Image Site: On the other hand, the sole purpose of an online art gallery is to deliver high quality images. These sites typically use thumbnails to deal with high access latencies. For our experiments, we used 4650 JPEG images totaling 480 Mbytes, downloaded from Photo.net [9].

3. Commerce Site: These sites would like to deliver large, high quality images to promote their merchandise without turning away users with high access latencies. For our experiments, we used 1248 JPEG images totaling 18 Mbytes, downloaded from Starwars.com [25].

For the JPEG images in the various workloads, we plot the cumulative distribution of the image size and the original JPEG Quality factor in Figure 2. From Figure 2(a), we note that images from PhotoNet are of high quality. Images in NCAR and NCSA have similar JPEG Quality factor distribution. From Figure 2(b), we note that size distributions are similar for Cnn, Starwars, NCSA and NCAR 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 distribution curve has knees at 10 KB and 100 KB.

In the interests 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.

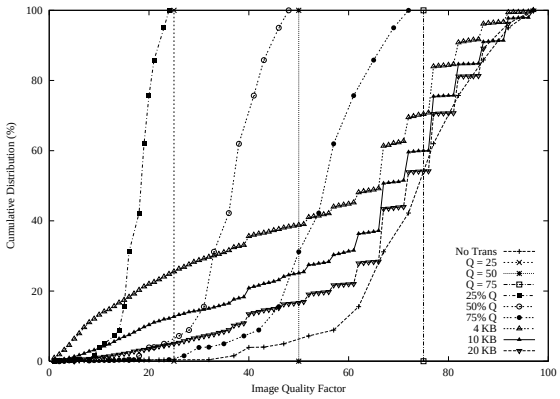
3.4.2 Transcoding Policies

For our experiments, we explored the following policies for transcoding JPEG images:

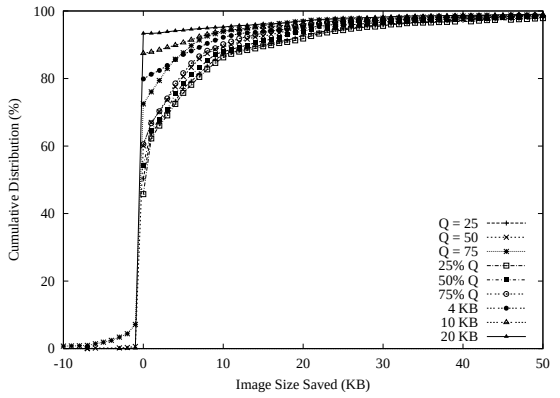
Simple: Following current practice, we transcode the JPEG images to ad hoc JPEG Quality factor values of 25, 50 and 75.

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 is 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 factor. We had experimentally verified that this linear assumption is valid for about 80% of the images. This assumption is not valid for large images; which make up a small percentage of the workload. Images that are smaller than this target size are not transcoded. This policy aggressively transcodes large files.

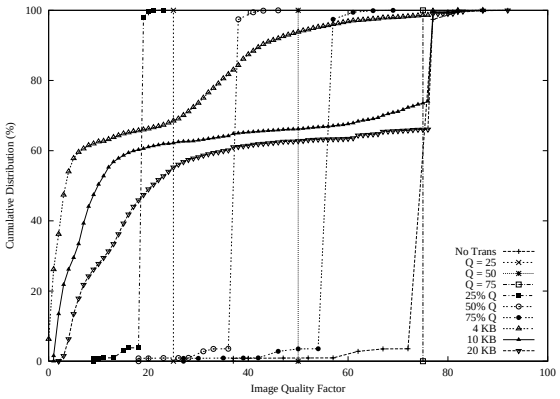


(a) Image Quality Distribution

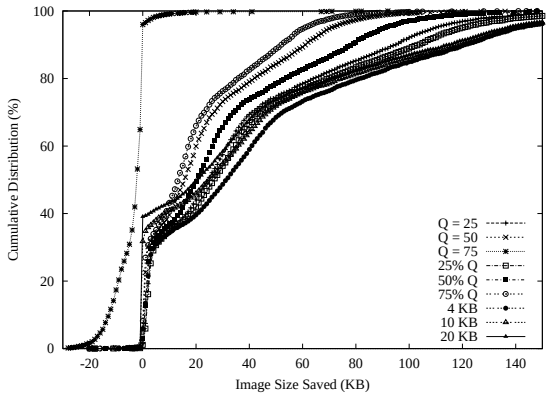


(b) Distribution of Image Size Saved (rounded down)

Figure 3: Perf. w/o knowledge of network characteristics (NCSA)



(a) Image Quality Distribution



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

Figure 4: Perf. w/o knowledge of network characteristics (PhotoNet)

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 be 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 [3] takes 7 $\mu secs$:

3.4.3 Implementation Details

For our experiments, we used a 450 MHz Pentium III with 512 MB of main memory, running FreeBSD 4.0. The various image collections were downloaded from the original web servers and stored on a dedicated 8 GB SCSI-2 disk. The proxy servers internal cache was stored in another separate, dedicated 8 GB SCSI-2 disk.

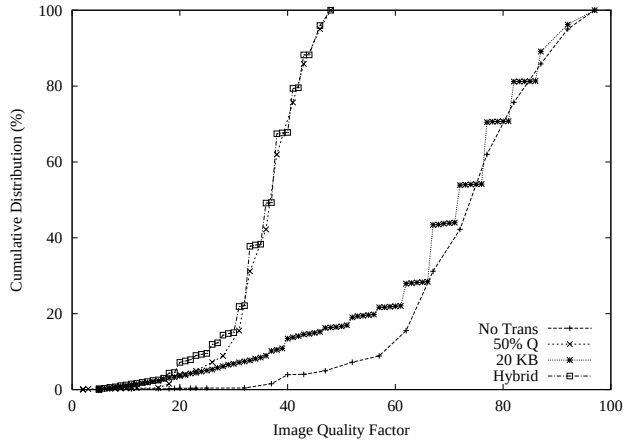
Proxy Server: For our experiments, we modified the Apache Server version 1.3.6 [2] to transcode JPEG images according to the various transcoding policies. For transcoding JPEG images, we used the IJG JPEG library. We used transcoding algorithms described in [3] to measure the initial JPEG Quality factor 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 proxy server serves every image using a new connection.

Client: We used http_load [23] to simulate accesses from clients using slow networks. http_load is a multi-processing http test client. We modified http_load to compute the individual access latencies as well throttle the access rates to 1000, 3360 and 6720 bytes per second, which corresponds to a 9600, 28800 and 56000 baud modems.

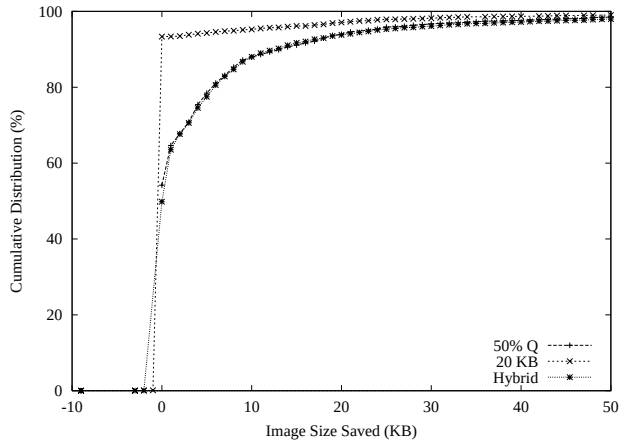
4 Results

4.1 Performance over a slow link without knowledge of network characteristics

The goal for this experiment is to measure the performance delivered to a client connected to its proxy server using a slow link. The proxies used in this experiment did not have knowledge of the network used by the client in accessing the proxy. Hence the same

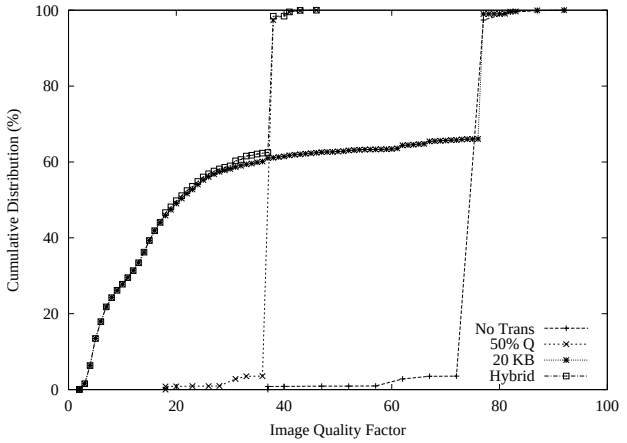


(a) Image Quality Distribution (NCSA)

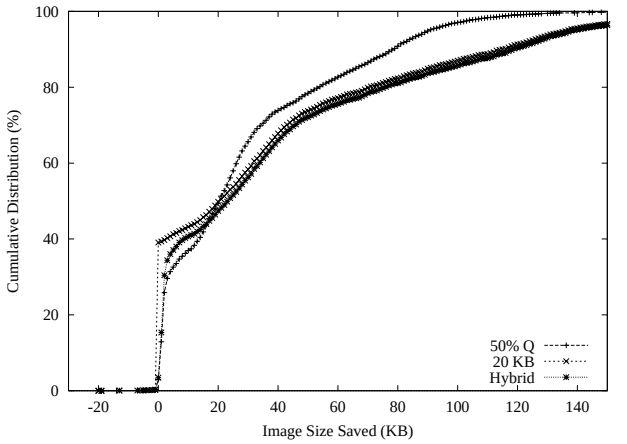


(b) Dist. of Image Size Saved (NCSA)

Figure 5: Perf. using a Hybrid Transcoding Policy (NCSA)



(a) Image Quality Distribution (PhotoNet)



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

Figure 6: Perf. using a Hybrid Transcoding Policy (PhotoNet)

image was served to clients, regardless of the network used in accessing the images.

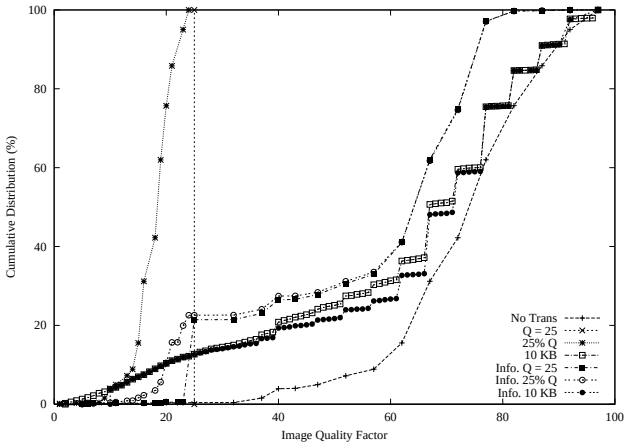
For the images in the NCSA and PhotoNet image collections, we plot cumulative distributions of the JPEG Quality factors of the transcoded images and the number of bytes not transferred in Figures 3(a) and 3(b) and Figures 4(a) and 4(b) respectively. . We want policies that can deliver images of Quality factors similar to the Quality factors provided by the non-transcoding server with savings in size.

We describe results for the NCSA and PhotoNet image collections. Results from performing experiments on the Cnn, Starwars and NCAR image collections were similar to the results from NCSA image collection and hence not described here.

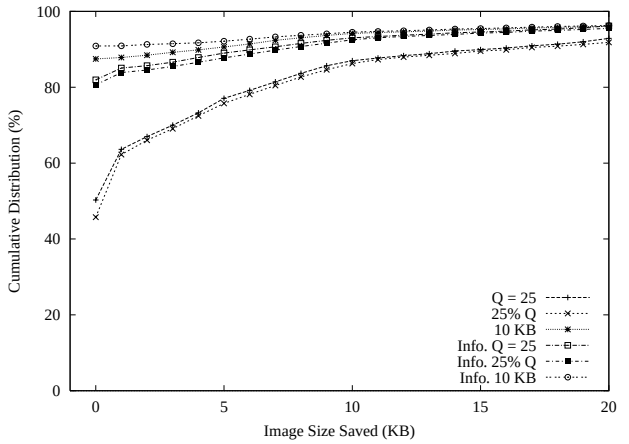
4.1.1 Policy Alternatives

Simple: First we analyze the results for the images in the NCSA image collection. From Figure 3(a), we note that simple policies transcode all images to a Quality factor of 25, 50 and 75 respectively. From Figure 2(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 Figure 3(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. These transcodings produce an image that is larger than the original image, obviously without added information content. From 3(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.

Next, we analyze the results for the images in the PhotoNet image collection. From Figure 2(a), we see that only about 5% of these images have a JPEG Quality factor less than 75. Similarly, Figure 4(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

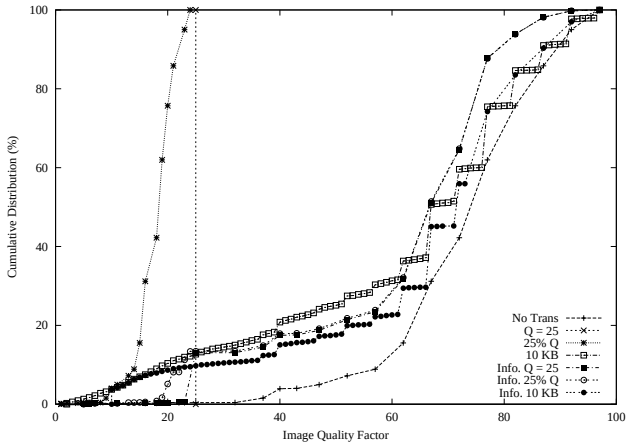


(a) Image Quality Distribution (9600)

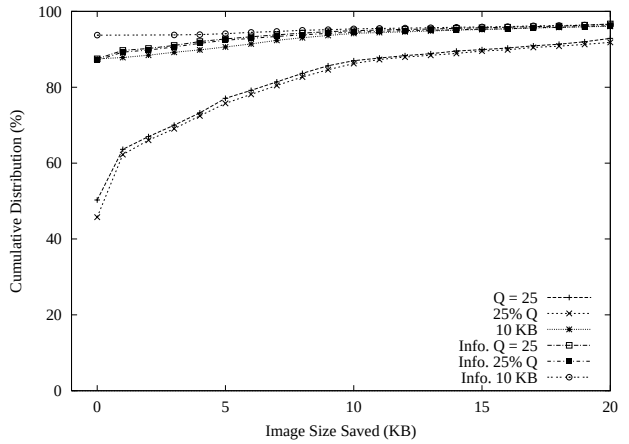


(b) Distribution of Image Size Saved (9600)

Figure 7: Server with knowledge of client link (NCSA; 9600)



(a) Image Quality Distribution (28800)



(b) Distribution of Image Size Saved (28800)

Figure 8: Server with knowledge of client link (NCSA; 28800)

size of up to 20 KB. For a policy that transcodes images to a Quality factor of 25, 50 and 75, about 10%, 10% and 40% of the images provided no savings in image size and about 40%, 50% and 5 % of the images in our collection saved from 0 through 60 KB respectively.

Relative quality: 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.

First we analyze the results for the images in the NCSA image collection. From Figure 3(a), we note that the JPEG Quality factor of the transcoded images is worse than the corresponding simple policy. From Figure 3(b), we note that for a transcoding that transcodes the images to Quality factor 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 in our collection saved from 0 through 10 KB respectively. These values are similar to the results for the simple policy.

Next we analyze the results for the images in the PhotoNet image collection. As with NCSA, Figure 4(a), shows that the JPEG Quality factor of the transcoded images is lower than the corresponding simple policy. From Figure 4(b), we note that for a transcoding that transcodes the images to Quality factor values of 25%, 50% and 75% of the initial JPEG Quality factor values, 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.

Target size: The target size policies transcode images to a target size such as 4 KB, 10 KB and 20 KB. They do not transcode images whose initial size is smaller than these values. Larger images are transcoded more aggressively than smaller images. This policy can be expected to provide image size benefits for large files.

First we analyze the results for the images in the NCSA image collection. From Figure 3(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 0-10KB, 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 results for the images in the PhotoNet image collection. Again the resulting Quality factors 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, 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. 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.

4.1.2 Hybrid Policies

Based on the results of performing our experiments on the PhotoNet and NCSA image collections we conclude that a transcoding policy that transcodes images to a target size aggressively transcodes large images generating big savings for large files, without any saving for 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 make it possible for the servers to make a choice on how aggressive 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 Figures 5 and 6 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.

4.2 Server with knowledge of client link characteristics

A proxy 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. For our experiments, we use ten seconds as the target client latency. Unnecessary transcodings not only reduce the Quality factor of an image, but can also place unnecessary CPU load on the proxy server.

Policy	Throughput (9.6K)		Throughput (28.8K)	
	Obj/sec	KB/sec	Obj/sec	KB/sec
No Transcoding	15.1	203	45.6	629
Q = 25	18.7	63	18.6	62
Q = 50	18.3	71	18.4	72
Q = 75	17.9	85	18.2	89
25% Q	17.7	58	19.0	60
50% Q	17.7	64	18.6	67
75% Q	17.6	70	18.3	75
4 KB	27.5	95	28.6	106
10 KB	31.9	150	37.0	184
20 KB	30.4	193	45.9	293
Info. 4 KB	32.3	149	44.4	292
Info. 10 KB	31.8	180	50.5	365
Info. 20 KB	28.4	197	54.99	451

Table 1: Computational Overhead (NCSA)

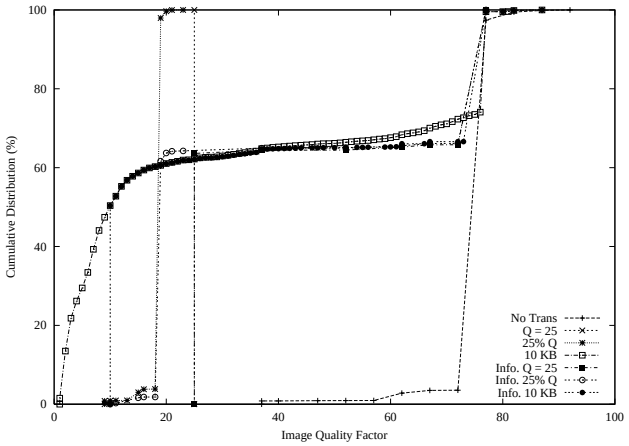
To better understand the server performance for the scenario where the transcoding proxy server knows the current network characteristics to the client, we perform experiments using informed transcoding for images in our image collection. The knowledge of the network characteristics are used by the transcoding proxy server to make an informed decision on whether transcoding is worth the effort. Our modified http_load client informs the proxy 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 fixed policy indicates otherwise. The informed policy transcodes images that are deemed to be worth the effort.

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.

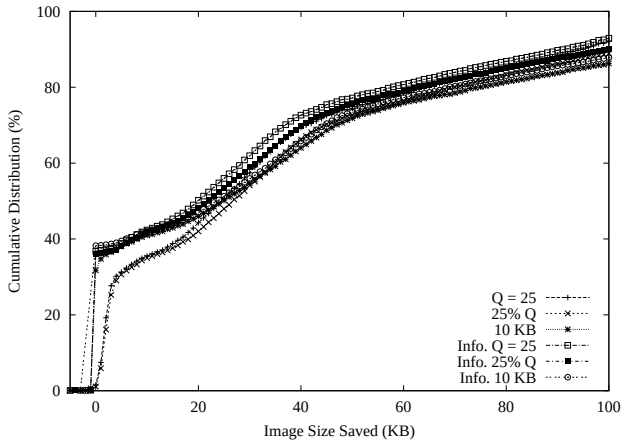
We measure the performance of the system using the measures of image Quality factor and the image size saved. We investigate the server CPU load savings in the next section. To reduce the complexity of the graphs, we illustrate the effects of informed transcoding for a policy that transcodes images to 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 3.4.2. We plot the image size saved and the image Quality factors as a cumulative distribution for clients accessing the proxy 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 Figures 7 through 10.

First we analyze the results for the images in the NCSA image collection. From Figures 7 and 8, we note that the informed policy deems 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

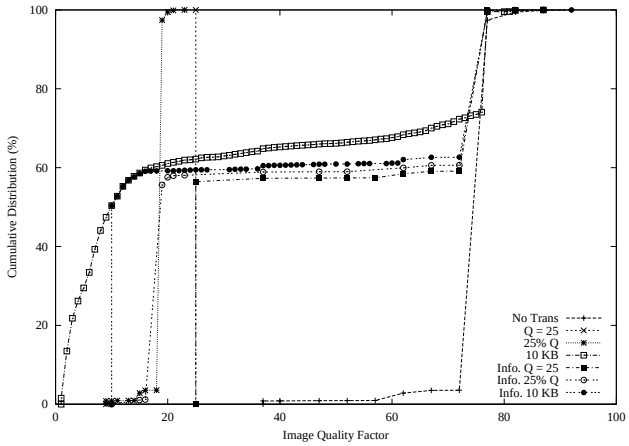


(a) Image Quality Distribution (9600)

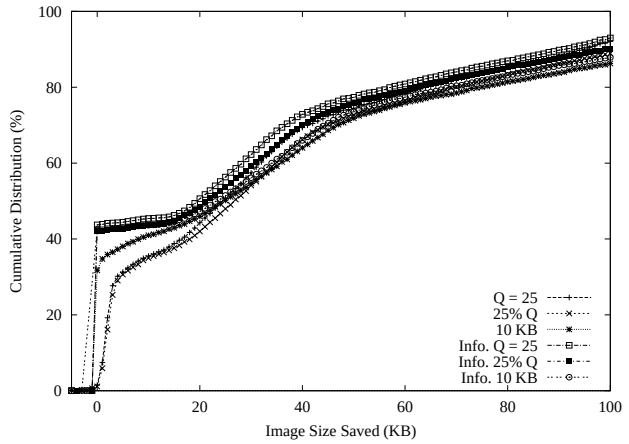


(b) Distribution of Image Size Saved (9600)

Figure 9: Server with knowledge of client link (PhotoNet; 9600)



(a) Image Quality Distribution (28800)



(b) Distribution of Image Size Saved (28800)

Figure 10: Server with knowledge of client link (PhotoNet; 28800)

transcodes 60% of the images, while the informed counterpart only transcoded 20% and 10% of the images for clients accessing the proxy server using a 9600 and 28800 baud modems respectively. A transcoding policy that transcodes images to a target size of 10 KB already did not perform transcoding for small files and hence informed policies did not make any difference for these images.

Next we analyze the results for the images in the PhotoNet image collection. From Figures 9 and 10, we note that with informed policies 35% and 42% of the images were not transcoded for clients accessing the proxy 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 proxy 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.

4.3 Computational overhead

In the last two sections, we analyzed different transcoding policies to determine whether informed transcoding offers better performance for the metrics of image information quality and size savings. Next, we analyze the computational overhead associated with implementing a transcoding proxy server that processes all images served anew rather than using the cache. Typical transcoding proxies can cache some of the transcoded images and hence provide even better service.

For our experiments, we use the NCSA image collections. We configured the Apache server to pre-fork 10 servers and serve up to 100 outstanding clients. Both the server and the http_load were executed on the same machine using the loopback network interface. Http_load throttles the rate at which it reads data to simulate accesses from slow clients. We tabulate the server throughput, measured in objects served per second and bytes served per second, using a 9600 baud modem and 28800 modem in Table 1.

A realistic proxy server would not only serve slow clients operating using wireless access technologies, but also serve clients accessing the server using fast LAN networks.

From Table 1, we note that for clients connecting to the server using a 9600 baud modem, the server can serve 15 objects per second at 203 KB/sec. We note that informed transcoding policies can provide better throughput than the uninformed case. Informed transcodings are needed for better server throughput when the clients connect over a faster modem such as 28800 baud.

We conclude that transcoding is a feasible operation on the proxy server that has a fast CPU. A commodity Pentium III processor can support informed transcoding operations and offer better service than a non-transcoding proxy server.

5 Conclusions and Future Work

In this paper, we explore an informed transcoding policy that utilizes knowledge of client link characteristics to maintain good quality of delivered multimedia content while maintaining predictable and user-settable access times. We further show that:

- Policies that transcode images to a fixed target size transcode large images aggressively and hence provide significant savings for large files. The transcoded images have JPEG Quality factors that closely match the base un-transcoded 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 can not only provide higher Quality factor images to the client, but also reduce the load on the proxy server by not performing unnecessary transcodings.

Acknowledgments

This work was supported in part by equipment grants from Intel Corporation and the National Science Foundation (CDA-95-12356).

References

[1] ALL THINGS WEB. Second state of the web survey (sows ii). www.pantos.org/atw/35654.html, May 1998.

[2] APACHE GROUP. Apache web server version 1.3.6. www.apache.org.

[3] CHANDRA, S., AND ELLIS, C. S. JPEG compression metric as a quality aware image transcoding. In *2nd Symposium on Internet Technologies and Systems* (Boulder, CO, October 1999), USENIX.

[4] CNN Interactive: All Politics. cnn.com/ALLPOLITICS/, December 1998.

[5] FLOYD, R., HOUSEL, B., AND TAIT, C. Mobile Web Access using eNetwork Web Express. *IEEE Personal Communications 5*, 5 (October 1998).

[6] FORD, A. M. *Relations between Image Quality and Still Image Compression*. PhD thesis, University of Westminster, May 1997.

[7] FOX, A., AND BREWER, E. A. 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.

[8] FOX, A., GRIBBLE, S. D., BREWER, E. A., AND AMIR, E. Adapting to network and client variability via on-demand dynamic distillation. *ACM SIGPLAN Notices 31*, 9 (Sept. 1996), 160–170. Co-published as SIGOPS Operating Systems Review **30**(5), December 1996, and as SIGARCH Computer Architecture News, **24**(special issue), October 1996.

[9] GREENSPUN, P. photo.net. www.photo.net, December 1998.

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

[11] HAN, R., BHAGWAT, P., LAMAIRE, R., MUMMERT, T., PERRET, V., AND RUBAS, J. Dynamic adaptation in an image transcoding proxy for mobile web browsing. *IEEE Personal Communications Magazine 5*, 6 (December 1998), 8–17.

[12] Intel quickweb. www-us-east.intel.com/quickweb/.

[13] JACOBSON, R. E., FORD, A. M., AND ATTRIDGE, G. G. 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).

[14] KANGASHARJU, J., KWON, Y., AND ORTEGA, A. Design and implementation of a soft caching proxy. In *3rd Intl. WWW Caching Workshop* (Manchester, England, June 1998).

[15] LANE, T., GLADSTONE, P., ORTIZ, L., BOUCHER, J., CROCKER, L., MINGUILLON, J., PHILLIPS, G., ROSSI, D., AND WEIJERS, G. The independent jpeg group's jpeg software release 6b. ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz.

[16] LILJEBERG, M., ALANKO, T., KOJO, M., LAAMANEN, H., AND RAATIKAINEN, K. 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).

[17] LILJEBERG, M., HELIN, H., KOJO, M., AND RAATIKAINEN, K. Mowgli www software: Improved usability of www in mobile wan environments. In *IEEE Global Internet 1996* (London, England, November 1996), IEEE Communications Society.

[18] MAZER, M. S., BROOKS, C., LOVERSO, J., THERAN, L., HIRSCH, F., MACRAKIS, S., SHAPIRO, S., AND ROCKWELL, D. Distributed clients for enhanced usability, reliability, and adaptability in accessing the national information environment. Tech. rep., The Open Group Research Institute, 11 Cambridge Center, Cambridge MA 02142, 1998.

[19] NOBLE, B. D., LI, L., AND PRAKASH, A. The case for better throughput estimation. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems* (Rio Rico, AZ, March 1999).

[20] NOBLE, B. D., SATYANARAYANAN, M., NARAYANAN, D., TILTON, J. E., FLINN, J., AND WALKER, K. R. Application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems and Principles* (Saint-Malo, France, October 1997).

[21] ORTEGA, A., CARIGNANO, F., AYER, S., AND VETTERLI, M. Soft caching: Web cache management techniques for images. In *IEEE Signal Processing Society 1997 Workshop on Multimedia Signal Processing* (Princeton NJ, Jun 1997).

[22] PENNEBAKER, W. B., AND MITCHELL, J. L. *JPEG - Still Image Data Compression Standard*. Van Nostrand ReinHold, New York, 1993.

[23] POSKANZER, J. Http load - multiprocessing http test client. www.acme.com/software/http_load/, 1998.

[24] SPECTRUM INFORMATION TECHNOLOGIES INC. Fastlane. www.spectruminfo.com.

[25] Star Wars - Episode I. www.starwars.com/episode-i/, December 1998.

[26] THE NATIONAL LABORATORY FOR APPLIED NETWORK RESEARCH. A distributed testbed for national information provisioning. http://ircache.nlanr.net/.

[27] TIMO, A., MARKKU, K., HEIMO, L., MIKA, L., MARKO, M., AND KIMMO, R. Measured performance of data transmission over cellular telephone networks. In *Computer Communications Review*, vol. 24:5. Computer Communications Review, 1994.