

# URICA: Usage-awaRe Interactive Content Adaptation for Mobile Devices

Iqbal Mohamed, Jim Chengming Cai, Eyal de Lara  
Department of Computer Science  
University of Toronto  
(iq, jcai, delara)@cs.toronto.edu

## ABSTRACT

Usage-awaRe Interactive Content Adaptation (URICA) is an automatic technique that adapts content for display on mobile devices based on usage semantics. URICA allows users who are unsatisfied with the system's adaptation decision to take control of the adaptation process and make changes until the content is suitably adapted for their purposes. The successful adaptation is recorded and used in making future adaptation decisions. To validate URICA, we implemented a prototype system called Chameleon that performs fidelity adaptation on web images. We conducted a user study in which participants used Chameleon to browse image-rich web pages on bandwidth-limited cellular links and used the collected traces to evaluate our system. We show that Chameleon reduces the latency for browsing web content by up to 65% and reduces bandwidth consumption by up to 80%. Chameleon also allows users to exchange bandwidth consumption for user interaction based on their personal preferences.

## Categories and Subject Descriptors

C.5 [Computer System Implementation]: Miscellaneous; H.4 [Information Systems Applications]: Miscellaneous

## General Terms

Algorithms, Performance, Experimentation, Human Factors

## Keywords

Content Adaptation, Mobile Devices, Customization, Learning

## 1. INTRODUCTION

Mobile devices face severe resource constraints, such as limitations in display size, battery life and network connectivity [22, 23, 35, 36]. In particular, the difference in bandwidth availability between desktop computers with broadband connectivity and mobile devices causes a significant degradation in the experience of mobile users as they go about using ubiquitous applications, such as browsing the World Wide Web. Even though the rollout of 2.5G

and 3G cellular technology has significantly increased the bandwidth that is theoretically available to mobile users, several recent studies have reported that, in practice, cellular bandwidth is considerably lower than originally claimed [4, 7, 8], and that the realistic throughput of wide-area mobile data services is similar to that of dial-up modems. For example, effective data rates for 2.5G technologies, such as GPRS and CDMA 1X, are reported at around 30-70 kbps and 100-120 kbps, respectively. In the experiments presented in this paper, where stationary users browsed the web on a PDA using CDMA 1X connectivity, we observed average bandwidth to be around 83 kbps. In addition, most reasonably priced data services charge based on the amount of bandwidth consumed by the user. A price of \$10 per MB of downloaded data is typical.

Some researchers have attempted to address the bandwidth constraint for mobile users by creating systems that distribute network communication over multiple cellular interfaces [33, 34]; however, this approach adds significant complexity and cost, which limits the deployment of such systems to specific usage scenarios, such as providing Internet connectivity to mobile users in a public bus via a specialized wireless hub that uses multiple cellular interfaces.

A promising alternative is to use automatic content adaptation systems that utilize lossy compression techniques to tradeoff content fidelity for bandwidth consumption [11, 14, 16, 24, 29, 31]. The key challenge in automatic content adaptation is the design of policies that perform appropriate adaptations. This is a hard problem because adaptation requirements often depend on the content's usage semantics. Consider the case of a web adaptation system that conserves bandwidth by reducing image fidelity. Suppose that an image of a 1964 Ford Mustang appears both on a Mustang Fan Club web page that includes images of many Mustang models from different years, and on a page of the Department of Transportation where the 1964 Mustang is used to illustrate what cars look like in general. It is likely that the fidelity level that would satisfy users of these two web sites would be different. Unfortunately, existing systems do not take usage semantics into account when making adaptation decisions. This results in adaptations that either waste system resources by loading higher fidelity content than required, or inconveniencing the user by providing objects at lower fidelity than what the user requires.

This paper introduces Usage-awaRe Interactive Content Adaptation (URICA<sup>1</sup>), an automatic technique that adapts content for display on mobile devices based on usage semantics. URICA allows users who are unsatisfied with the system's adaptation decision to take control of the adaptation process and make changes until the content is suitably adapted for their purposes. The successful adaptation is recorded and used in making future adaptation decisions for the *same* or *other* users. URICA is a general adaptation tech-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*EuroSys '06*, April 18–21, 2006, Leuven, Belgium.

Copyright 2006 ACM 1-59593-322-0/06/0004 ...\$5.00.

<sup>1</sup>pronounced Eureka

nique that can be applied to a wide range of problems, such as fidelity adaptation of images and video, customizing the layout of HTML documents for display on small screens, or automatic user interface repurposing for heterogeneous devices.

We also present Chameleon, a system that applies the URICA technique to the problem of adapting image-rich web pages for browsing over bandwidth-limited network links. Chameleon reduces the latency and bandwidth consumption of browsing web pages by loading images at lower fidelity. When the system's prediction results in serving an image at lower fidelity than what the user requires, Chameleon lets the user interact with the image to increase its fidelity. Chameleon learns from the user's feedback and adjusts its prediction for future accesses to the same image. For example, Chameleon initially serves the image of the 1964 Mustang to users of the two web sites described above at the same low fidelity level; however, after observing that users of the Mustang Fan Club page increase the image's fidelity, Chameleon adjusts its prediction and starts serving users of this page higher fidelity versions of the image. Users of the Department of Transportation page continue to get the lower fidelity version.

By default, Chameleon minimizes the latency for browsing web pages; however, Chameleon can also be configured to allow the exchange of bandwidth consumption, which for many cellular users has a direct monetary cost, for user involvement in the adaptation process. In this mode, the user can specify the minimum expected monetary savings that would make it worthwhile for her to provide interaction. If a user prefers to save money, Chameleon becomes more aggressive and serves images at lower fidelities. This saves bandwidth, and thus money, but increases the likelihood that the user will have to interact with the system to obtain a satisfactory adaptation. Alternatively, if a user prefers to interact with images infrequently, the savings she will require for every interaction will be greater, and Chameleon will serve images at higher fidelities. This reduces the amount of interaction that is required but can cause more bandwidth consumption as the system may serve images at fidelities greater than what the user actually needs.

We evaluated Chameleon in a controlled user study where 30 University students were asked to use Chameleon to browse image-rich web content. Chameleon presented users with highly adapted images and allowed them to improve image fidelity one level at a time. The user study validated Chameleon's assumptions that content adaptation requirements depend on usage semantics. When performing the same tasks, we found that users shared adaptation requirements for the same images but the desired adaptation varied from image to image. Also, when images were used to complete different tasks, we found that fidelity selections were clustered around a small set of clearly identifiable fidelity levels.

Chameleon reduces latencies of browsing web content by up to 65% and can reduce the amount of bandwidth consumption by as much as 80%. It achieves large savings even when content is used for multiple tasks that have different adaptation requirements. Chameleon is robust and provides consistently good performance across different web sites and over different network conditions.

This paper makes four contributions: (i) it introduces URICA, an automatic adaptation technique that learns how to adapt content for display on mobile devices by allowing users to interact with the adaptation system to correct bad adaptation decisions; (ii) it shows that content adaptation requirements depend on usage semantics, and that users browsing the same content share adaptation requirements; (iii) it shows that minimizing the latency of browsing adapted content requires taking into account the relative latencies associated with data transfer and user interaction to correct bad adaptation decisions, and provides an algorithm that minimizes the overall

latency; and (iv) it shows that it is possible to exchange bandwidth consumption for user interaction, and provides an algorithm that lets users control this tradeoff.

The rest of the paper is organized as follows. Section 2 introduces Usage-aware Interactive Content Adaptation (URICA). Section 3 describes Chameleon, a URICA prototype, that adapts image-rich web pages for browsing over bandwidth-limited network links, and Section 4 details the workings of Chameleon's prediction algorithms. Section 5 presents the user study we conducted to collect traces of users browsing adapted image-rich web content. Section 6 evaluates the performance of Chameleon. Finally, Section 7 discusses background material on content adaptation and compares our work with previous efforts, and Section 8 presents our conclusions and avenues for future work.

## 2. USAGE-AWARE INTERACTIVE CONTENT ADAPTATION (URICA)

Usage-aware Interactive Content Adaptation (URICA) is an automatic content adaptation technique that takes usage semantics of content into account when making its adaptation decisions. URICA empowers users by allowing them to change adaptation decisions made by the system. If a user determines that the adaptation performed on content is not satisfactory, she can instruct the application to correct the situation (e.g., make content more readable by removing an irrelevant toolbar, convert text to speech, or increase a video's frame rate). URICA views user modifications of an object's adaptation as corrective feedback on the system's adaptation decision, and uses these corrections to improve the quality of adaptation predictions for future accesses to the object by the *same* or *other* users. Thus, URICA adaptation decisions reflect the way users utilize content, in effect capturing the content's usage semantics. URICA is a general concept and can be applied to a wide range of adaptation problems, such as fidelity adaptation of images, audio and video, page-layout customization for display on small screens, or automatic user interface repurposing for heterogeneous environments.

Adaptation systems based on URICA consist of three components: *client applications*, an *adaptation proxy* and *content servers*. Client applications present adapted content to users and allow them to modify the decision made by the adaptation system. The adaptation proxy mediates all accesses to content, provides adapted content, maintains history, and serves as an aggregation point for user requests. Content servers are standard unmodified data repositories, such as web sites, databases and media servers, and no additional functionality is required for this component.

In the rest of this section, we first describe the support required from client applications and the requirements of the adaptation proxy. We then summarize the operation of the adaptation system and discuss implementation issues.

### 2.1 Client Application Support

Client applications allow users to interactively change adaptation decisions to make content more appropriate for their tasks. A client application has to provide the following functionality: (1) it should make the user aware that the content has been adapted (or could be adapted), (2) it should provide an appropriate user-interface for modifying adaptation decisions, (3) it should propagate the user's adaptation corrections to the adaptation proxy, (4) for adaptation corrections that require the adaptation proxy to supply a different version of the content (e.g., a higher-fidelity version of an audio file), the application has to support fetching of the new version, possibly in the form of a delta, and updating the application's state

accordingly <sup>2</sup>, and (5) the application can provide an interface for users to specify adaptation preferences (e.g., system should minimize bandwidth consumption).

The interface that the client application provides for correcting adaptation decisions depends on the datatype being adapted, and the supported adaptations. For example, an application that supports adapting streaming video may provide a user-interface to modify the size of the window that plays the video, the video's resolution, frames per second, or the position of the stream where playback starts. For document-layout adaptation, the application can provide the ability to split content into panels or separate windows, rearrange content position, or remove parts of a document.

In most cases, adding feedback mechanisms such as those described above requires modifications to the source code for the application. Alternatively, the increasing popularity of component-based software allows the feedback mechanism to be incorporated into a plug-in component, which is developed using an application's API [12].

## 2.2 Adaptation Proxy Requirements

The adaptation proxy mediates all of the client's content requests. The proxy performs the following tasks: (1) determines the adaptation to perform based on predictions, (2) retrieves the original content from content servers and adapts as needed, (3) serves the adapted content to the client, (4) responds to client requests to change the adaptation to perform on content, and (5) keeps track of user adaptation corrections (feedback).

The adaptation proxy accumulates user feedback at the granularity of individual content items (e.g., individual images in an HTML page). The proxy relies only on feedback collected in the course of normal application usage. The proxy assumes that the prediction failed when the user changes the way the content was customized (e.g., the user increases the fidelity of an image, or resizes a text box). Conversely, the proxy assumes that an appropriate customization has been found when the user opts not to change the content any further. Therefore, the proxy views user modifications as negative feedback (i.e., something was wrong with the adaptation), and the lack of user action as positive feedback (i.e., the user was satisfied with the adaptation decision). An alternative is to explicitly ask users for feedback. This approach, however, results in significant burden as users have to provide additional information that is not relevant to the performance of their task.

## 2.3 Adaptation Process

The adaptation process in URICA-based systems operates as follows. A user first requests content through the client application. The client application forwards the user's request to the adaptation proxy. The adaptation proxy makes an adaptation decision based on the history of adaptations that satisfied other users in the past. <sup>3</sup> Once the adaptation decision is made, the adaptation proxy checks whether it has an appropriately adapted version of the desired content. If so, the adapted content is served to the user. If not, the content is first fetched from the content server, and then it is appropriately adapted and cached. If the user is unhappy with the version of the content that she got, she can then use the interface provided by the application to apply a corrective measure. The user's correction is then sent to the adaptation proxy where it becomes part of

<sup>2</sup>Not all adaptation corrections involve the transmission of a new content version. For example, an application that supports page-layout adaptation may be able to locally support the reorganization of the elements within a document.

<sup>3</sup>When no history is available, a set of default rules or constraints can be used to make the adaptation decision.

the history of the particular object being adapted and affects future adaptation decisions. If the corrective measure requires the transmission of data from the adaptation proxy (such as a higher fidelity image), the content (potentially in the form of a *delta*) is sent to the application and the application updates its state. In the case of modifications to streamed data, the client application and the adaptation proxy negotiate to change the necessary properties of the stream.

## 2.4 Discussion

URICA requires a critical mass of viewers of the same content. At first glance, this prevents the applicability of URICA to adapting dynamic Web content. Closer examination reveals that even in cases where significant reuse of content is not apparent, such as with dynamic content, URICA can prove useful. First, many dynamic web pages are built from a set of static objects (such as images) that are served to many users. Here, the scenario is no different because the user feedback accumulates for individual objects that do not change. However, there are situations where content is dynamically created for a specific user (e.g., an image showing the current prices of stock in a user's investment portfolio). If content is created dynamically and tailored to a specific user, it is unlikely that we can accumulate history for that object. In this situation, the system can attach history to a placeholder representing the dynamic content if we expect that the adaptation decisions carry over across instances of the dynamically generated content. However, if a user is the sole viewer of dynamic content that bears no resemblance to past instances, then URICA will not apply as meaningful history cannot be accumulated.

To improve performance, recently requested objects and their adapted versions can be cached on the adaptation proxy. It should be noted, however, that objects will typically be cached for much shorter durations (minutes or hours) than their history (days, weeks or more).

Most of the complexity inherent in the URICA technique is relegated to the resource-rich servers hosting the adaptation proxies. Therefore, mobile clients are shielded from the resource intensive tasks associated with storing large history logs, running the prediction mechanisms, and performing transcoding.

User privacy is likely to be a concern in URICA. Whereas many other systems can track the data that was accessed by a user, systems based on URICA can actually discern which objects captured the user's attention, and therefore are more relevant. This issue can be addressed by having anonymous history traces and by aggregating user requests. An evaluation of the tradeoffs related to tracking of individual user history versus aggregates, and how these aspects influence the quality of predictions is beyond the scope of this paper.

## 3. CHAMELEON

Chameleon is a URICA prototype that supports fidelity adaptation for browsing image-rich web pages over bandwidth-limited network links. Chameleon provides fidelity-adapted versions of web images but allows users to interact with individual images in order to improve their fidelity.

Chameleon consists of an adaptive web browser running on an AudioVox PDA with built-in CDMA 1X cellular connectivity, and an adaptation proxy that performs progressive JPEG image transcoding. The proxy runs on a well-provisioned server with a high-bandwidth and low-latency connection to the Internet. The PDA connects to the Internet over the cellular CDMA 1X link, but is configured to route all web traffic through the adaptation proxy.

By default, Chameleon minimizes the latency for browsing adapted web pages. Specifically, Chameleon optimizes a user's *fulfillment time*, which is the sum of the download time to transfer content (including refinements) and the user interaction time required to correct bad adaptation decisions. Chameleon can also be configured to allow the exchange of bandwidth consumption, which for many cellular users has a direct monetary cost, for user involvement in the adaptation process. Specifically, a user can specify the minimum expected monetary savings that would make it worthwhile for her to provide interaction.

In this section, we describe the adaptive browser and adaptation proxy used in Chameleon. We defer discussion of the adaptation proxy's prediction mechanism until Section 4.

### 3.1 Adaptive Browser

The adaptive browser is based on the Pocket PC version of Internet Explorer. To allow users to refine images, we augmented Internet Explorer with two user interfaces. We will refer to these interfaces as *tapping* and *multi-tab*. With the tapping interface, users can improve the fidelity of an image by tapping on it with the stylus. With the multi-tab interface, users can increase image fidelity by multiple levels in a single operation. The multi-tab interface shows the available fidelity levels for an image with an array of small boxes above the image. To request a specific fidelity level, the user just taps on the appropriate box with the stylus. Boxes with unavailable fidelity levels, such as those at or below the current level, are grayed out. Figure 1 shows a screenshot of the multi-tab interface. We implemented two interfaces to allow us to research the effects that different feedback mechanisms have on the quality of adaptation predictions. However, determining the best interface for applications that support fidelity adaptation is beyond the scope of this paper.

If Chameleon is being used to optimize the user's fulfillment time, the adaptive browser must determine the user's average *interaction time* and relay this back to the adaptation proxy. We define a user's interaction time to be the time it takes her to determine that the current adaptation is inappropriate and request an improvement. We acknowledge that determining interaction time is user-interface specific. For instance, the computation of interaction time in an interface that allows the fidelity of images to be improved individually will be different from that in an interface that permits the simultaneous refinement of all of the images on the screen or page. In our implementation of the tapping and multi-tab interface, when there is a single image on the page, average interaction time is simply the time between when the image is rendered on the screen and when the user provides an interaction, averaged across many instances of interaction. However, when a page has multiple images, we only consider the time between consecutive interactions on the same image, and disregard the time if the user provided interaction to a different image in between.

The adaptive browser also provides a user-interface that allows users to specify whether they are interested in optimizing fulfillment time or whether they want to achieve a desired tradeoff between bandwidth consumption and the need to provide interaction. When the user wishes to exchange bandwidth for interaction, the interface lets the user specify the minimum expected monetary savings that would make it worthwhile for her to provide interaction. Also, if the adaptation proxy cannot automatically determine the user's monetary cost of bandwidth, the user can provide this information. The browser relays this information to the adaptation proxy.



**Figure 1: Snapshot of the PDA with the multi-tab interface that allows users to select an arbitrary fidelity level. The left and right snapshots show the same image at low and high fidelity, respectively.**

### 3.2 Adaptation Proxy

In response to a request from the client, the proxy downloads the web content from the origin web server (which is unmodified and is not aware that content is being adapted) and transcodes any images into a progressive JPEG format. The proxy then serves the HTML file to the client along with fidelity-adapted versions of all images. We configured the proxy to generate progressive JPEG images consisting of 10 scans. Progressive JPEG images have the property that if we only have the first few scans of the image, we can see a low fidelity version of the image and as we load subsequent scans, image quality improves. The relationship between fidelity levels and scans is that fidelity level  $i$  corresponds to the progressive JPEG image consisting of the first  $i$  scans of the image. Users can refine the fidelity of individual images by requesting additional scans.

A key function of the adaptation proxy in the URICA model is to keep track of the history of adaptations for an object that satisfied users in the past. The proxy assumes that the highest fidelity of an image that is served to a user is the adaptation that provides satisfaction and we call this fidelity level the *Fulfillment Fidelity*. For each image, the adaptation proxy keeps track of histories of fulfillment fidelities of different users. The history of successful adaptations is used by our prediction policies, which we describe in detail in Section 4.

#### 3.2.1 Learning and Prediction Modes

Chameleon only considers implicit feedback when recording the history of fulfillment fidelities. That is, when an image is served at a certain fidelity and the user does not provide any interaction, Chameleon assumes that this is the fulfillment fidelity of the user. Chameleon knows that the user was able to perform the task, but Chameleon cannot assume the optimality of the adapted version. For example, serving an image at a higher fidelity than required for a given task will enable the user to perform the task, but will waste resources. While the transmission of extra data forces the user to wait longer, it is unlikely that the user, having waited for the higher-resolution object to load, will instruct the application to lower the image's resolution. While the user must interact with the system to correct adaptation decisions that prevent her from carrying out her task, there is little motivation for modifying adaptations that enable the task but waste resources. This is particularly true given that the

resources would have already been wasted (i.e., the high-fidelity image was already loaded) and that interaction requires time and effort <sup>4</sup>.

We address the ambiguity of implicit feedback by having two serving modes for every object: *Learning* and *Prediction*. The purpose of the learning mode is to allow the system to rapidly attain high quality histories of fulfillment fidelities and the duration of this mode,  $L$ , is specified by the system operator. For each image, the first  $L$  users who access it are not provided with any prediction. Instead, the proxy always serves the first scan of the image initially, i.e., fidelity 1. After the first  $L$  accesses, the system switches into prediction mode. In prediction mode, the system uses prediction algorithms, described in the next section, to optimize a user’s fulfillment time or a specified tradeoff between bandwidth and the number of interactions. In practice, the usage semantics of content may change over a long period of time. To address this, it may be desirable for the system to periodically discard all accumulated history for an object and return to learning mode. This would ensure that the system’s predictions utilize the current usage semantics of the content. An investigation of the impact of changing usage semantics over time is out of the scope of this paper.

Regardless of whether the tapping or multi-tab interface is being used, images are initially served at the lowest fidelity in learning mode. In contrast, in prediction mode, the initial fidelity of served images is determined by a prediction. With the tapping interface, tapping on an image in learning mode increases its fidelity level by one; however, tapping on an image in prediction mode provides the next prediction, if available. Otherwise, the fidelity of the image is improved by one. With the multi-tab interface, after the image is initially served, the user can request arbitrary improvements to the image’s fidelity. However, in prediction mode, if there are subsequent predictions available, these are highlighted in the user-interface.

## 4. CHAMELEON PREDICTION POLICIES

In this section we discuss two types of history-based prediction policies in Chameleon. First, we describe prediction policies based on statistical functions applied to the history of fulfillment fidelities. Next, we describe the Personalized Adaptation Schedule (PAS) prediction algorithm. PAS bases its adaptation decision on the history of fulfillment fidelities, and also considers user-specific information such as the user’s preference regarding the exchange of bandwidth for interaction and average interaction time.

### 4.1 Statistics-Based Policies

We consider various statistics-based policies for predicting the fidelity level at which an image should be served: MEAN, MEDIAN, MODE and MAX. These policies select the mean, median, mode and maximum of the entries in the history of fulfillment fidelities, respectively. MEAN, MEDIAN and MODE are measures of central tendency in a distribution and the prediction made by the policies will reflect the fidelity levels most desired by users. MAX, on the other hand, attempts to minimize the need for users to provide interaction by serving objects at the highest fidelity level that was encountered so far.

Statistics-based policies make a single adaptation decision. If the predicted fidelity is less than the user’s fulfillment fidelity, the user must interact with the image in order to improve its fidelity.

<sup>4</sup>For some data types such as streaming video, it may be the case that a user conscious of her power or bandwidth usage may be willing to interact with the system to explore whether less resource-intensive versions to the one currently being streamed meet her task requirements.

## 4.2 Personalized Adaptation Schedule (PAS)

Personalized Adaptation Schedule (PAS) is a prediction algorithm that takes into account the possibility that the user may be unsatisfied with its previous prediction and would have to provide interaction. Variations among individual users, as well as the fact that the same image sometimes has multiple usage semantics (i.e., the image is being used for multiple tasks) implies that a single adaptation decision may not satisfy all users. PAS addresses this by creating a list of fidelity predictions, where the first fidelity in the list is served to the user initially and the following fidelities are served upon subsequent interactions. We call this list of predictions a *prediction schedule*. The prediction schedule is individualized for every object and user but it has to be computed only once (when an object is first accessed by a user). PAS has two modes of operation. In the first mode, PAS optimizes a user’s fulfillment time, which is the sum of the download time to transfer content (including refinements) and the user interaction time required to correct bad adaptation decisions. The second mode allows users to specify a tradeoff between bandwidth and user interaction, and PAS optimizes for this. In the rest of this section, we first describe how PAS creates prediction schedules that optimize fulfillment time, and then show how PAS can create prediction schedules that achieve a desired tradeoff between bandwidth consumption and user interaction.

### 4.2.1 Fulfillment Time Optimization

PAS can be used to create prediction schedules that optimize a user’s expected fulfillment time. In Chameleon, fulfillment time is the sum of download times for successively higher fidelities of an image (which depends on image file-size, bandwidth and latency) and the user interaction time that is required to request refinements. PAS balances these two components so that the user does not have to spend excessive time downloading data over the network and also does not waste too much time interacting with the system.

We start by providing an analytical model for determining the user’s expected fulfillment time. This model is used by PAS to create a prediction schedule that optimizes a user’s expected fulfillment time. In our model, we suppose that a progressive JPEG image is broken into  $n$  slices. We label the slices  $s_1, \dots, s_n$ . Let fidelity level  $i$  correspond to the image composed of all slices from  $s_1$  to  $s_i$ .<sup>5</sup> Let  $ff$  be the user’s true fulfillment fidelity, which is unknown a priori, and thus a random variable in the model. Next, let  $I$  be the current user’s average interaction time. Let  $DT(i, j)$  be the time required to download the slices  $s_{i+1} \dots s_j$ .

We can now define  $ET(i, j)$  (where  $0 \leq i < j \leq n$ ) to be the user’s expected fulfillment time when they request a refinement at fidelity level  $i$  (the user’s current fidelity level) and we decide to take them to some arbitrary fidelity level  $j$ .<sup>6</sup> The way in which we calculate the value of expected fulfillment time depends on whether or not we have provided the user with the highest fidelity level for the image (i.e. whether or not  $j = n$ ). If  $j = n$ , we know that the user’s fulfillment fidelity ( $ff$ ) cannot be higher than  $j$  since  $n$  is the highest fidelity level, and the user will not request any refinements subsequently. In this case, the user’s expected fulfillment time is simply the download time required to transfer slices  $s_{i+1} \dots s_n$  of the object. However, if  $j < n$ , fidelity level  $j$  may be below the user’s true fulfillment fidelity and the user may need to refine the image again. The user requests further refinements with  $Prob(ff > j | ff > i)$ , which is the probability that when the user

<sup>5</sup>We add an additional definition to simplify the presentation of our equations below. We define fidelity level 0 to be the case when the user has slice  $s_0$ , which is empty. This corresponds to the initial situation when the user does not have any data.

<sup>6</sup> $j$  does not have to equal  $ff$

is at fidelity level  $j$ , they need further refinements conditioned on the fact that the user's fulfillment fidelity level is greater than  $i$ . We assume that if further refinements are required, we will serve the user with the fidelity level  $j'$  that minimizes her expected fulfillment time. Thus, the user's expected fulfillment time in the case where  $j < n$  has two components:  $DT(i, j)$  and, if  $j < ff$ , the time cost of providing an interaction summed with the the optimal expected fulfillment time starting from  $j$ . These two cases are defined in the following recurrence relation:

Base case:

$$ET(i, n) = DT(i, n) \quad (\text{where } 0 \leq i < n)$$

Recursive case:

$$ET(i, j) = DT(i, j) + Prob(ff > j | ff > i) \times (I + \min[ET(j, j+1), \dots, ET(j, n)]) \quad (\text{where } 0 \leq i < j < n)$$

We can now describe how PAS creates a prediction schedule, which is a list of fidelities, where the first fidelity in the list is served to the user initially and the following fidelities are served upon subsequent interactions. To create the schedule, we first solve the recurrence  $ET(i, j)$  for all pairs of fidelity levels. This can be easily done in polynomial time using dynamic programming techniques to avoid unnecessary recomputations. Once computed, these expected values are stored in a table for quick access. Next, we find the fidelity level  $a$  such that  $ET(0, a) = \min[ET(0, 1), \dots, ET(0, n)]$ . This becomes the first prediction in our schedule. Whenever we have a prior predicted fidelity level, say  $i$ , we can find the subsequent prediction  $j$  such that  $ET(i, j) = \min[ET(i, i+1), \dots, ET(i, n)]$ . This process is repeated until the prediction made is fidelity level  $n$ .

Figure 2 demonstrates a simple example of expected fulfillment time computations. This figure is divided into 3 columns. The first column shows past history, which is used to form probabilities. The center column provides the calculations of expected fulfillment time (ET) for all possible  $i, j$  combinations. Finally, the rightmost column performs the computation for an illustrative set of parameters. In our example, we set the object to be divided into  $n=4$  slices, interaction time  $I$  to be 1 second, and the time cost of transferring a single slice to be 1 second also. Once all the expected fulfillment time values for this user have been computed, we can compute the prediction schedule for this user as follows: We suppose that the user currently has the image at fidelity 0 (i.e. the user does not have the image). We consider all possible adaptation decisions at this level and pick the one that provides the lowest fulfillment time to be the first prediction in our schedule. From  $ET(0,1)$ ,  $ET(0,2)$ ,  $ET(0,3)$  and  $ET(0,4)$ , we see that the expected fulfillment time for providing fidelity 2 is the lowest (3.50 vs. 3.66, 4.00 and 4.16 seconds), and so this is the first prediction in our schedule. To compute the next prediction, we suppose that the user sees the image at fidelity 2 and is not satisfied. We again consider all possible adaptation decisions at this level and pick the one that provides the lowest fulfillment time to be the second prediction in our schedule. From  $ET(2,3)$ , and  $ET(2,4)$ , we see that the expected fulfillment time for providing fidelity 4 is the lowest (2.00 vs 2.33 seconds), and so this becomes the second prediction in our schedule. Since a prediction in our schedule has reached the highest fidelity ( $n=4$ ) in our example, we are done. The computed schedule consists of two predictions: 2 and 4.

#### 4.2.2 Achieving a Desired Tradeoff between Bandwidth Consumption and User Interaction

Often, it is the case that users have to pay for the amount of bandwidth that they use over wide-area cellular links. An adaptation system that minimizes bandwidth consumption, which is how our system behaves in Learning mode, would provide users with the minimal amount of bandwidth that satisfies them and wastes no unnecessary bandwidth. However, this requires the maximum number of interactions and users may find this cumbersome. Our system allows the user to specify a tradeoff between their willingness to interact and their desire to save bandwidth (and thus money). Specifically, a user can specify the minimum expected monetary savings that would make it worthwhile for her to provide interaction. We now show how PAS can optimize a user-specified tradeoff, taking into account both the monetary cost of interaction and the price of bandwidth.

Let  $EC(i, j)$  (where  $0 \leq i < j \leq n$ ) be the user's expected monetary cost if, given that the user has fetched all fidelity levels up to level  $i$  and is requesting a fidelity improvement, we take them to fidelity level  $j$ . Let  $BC(i, j)$  be the bandwidth cost required to download the slices  $s_{i+1} \dots s_j$ . This is just the cost per KB of data multiplied by the sizes of all the slices that need to get transferred. Finally, let  $IC$  be the minimum cost savings that the user requires in exchange for providing a single interaction. Now, we can define the following recurrence relation:

Base case:

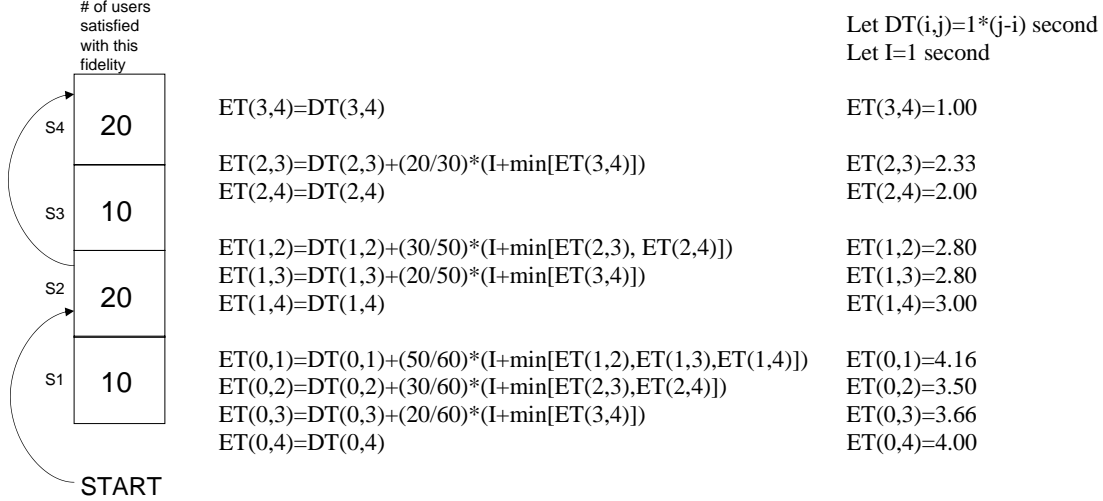
$$EC(i, n) = BC(i, n) \quad (\text{where } 0 \leq i < n)$$

Recursive case:

$$EC(i, j) = BC(i, j) + Prob(ff > j | ff > i) \times (IC + \min[EC(j, j+1), \dots, EC(j, n)]) \quad (\text{where } 0 \leq i < j < n)$$

The base case is the user's expected monetary cost if we take them to fidelity level  $n$  when they requested an improvement at fidelity level  $i$ . This is simply the bandwidth cost of transferring slices  $s_{i+1} \dots s_n$  of the object. The recursive step is the user's expected monetary cost if we take them to fidelity level  $j$  when they requested an improvement at fidelity level  $i$ . The expected monetary cost in this case is the bandwidth cost of downloading the delta between  $i$  and  $j$ , along with the cost of providing interaction and the bandwidth cost of downloading additional slices if fidelity  $j$  was insufficient. The monetary cost of further adaptation is only incurred with  $Prob(ff > j | ff > i)$ , which is the probability that when the user is at fidelity level  $j$ , they need further refinements (conditioned on the fact that the user's fulfillment fidelity level is greater than  $i$ ).

We can now describe the workings of the PAS algorithm that creates a prediction schedule to minimize the expected monetary cost of adaptation and thus achieving the user-specified tradeoff between bandwidth consumption and user interaction. To create the schedule, we first solve the recurrence  $EC(i, j)$  for all pairs of fidelity levels. As before, this can be easily done in polynomial time using dynamic programming techniques to avoid unnecessary recomputations. Once computed, these expected values are stored in a table for quick access. Next, we find the fidelity level  $a$  such that  $EC(0, a) = \min[EC(0, 1), \dots, EC(0, n)]$ . This becomes the first prediction in our schedule. Whenever we have a prior predicted fi-



**Figure 2: Example of expected fulfillment time computations. The left column shows past history, which is used to form probabilities. The center column provides the calculations of expected fulfillment time (ET) for all possible  $i, j$  combinations. The rightmost column performs the computation for an illustrative set of parameters. The computed schedule consists of two predictions: 2 and 4.**

delity level, say  $i$ , we can find the subsequent prediction by finding the  $j$  such that  $EC(i, j) = \min[EC(i, i + 1), \dots, EC(i, n)]$ . This process is repeated until the prediction made is fidelity level  $n$ .

## 5. EXPERIMENTS

We conducted a user study to collect traces of users interacting with adapted image-rich web content using our Chameleon prototype. The objective of the experiment is to validate the following hypotheses:

- $H_1$ : Fidelity adaptation can significantly reduce bandwidth consumption.
- $H_2$ : Users with the same usage semantics have similar adaptation requirements.
- $H_3$ : Adaptation requirements vary between images.
- $H_4$ : Users with different usage semantics have different requirements for adaptation.
- $H_5$ : Usage semantics affect adaptation requirements.
- $H_6$ : Prediction based on history will result in good adaptation performance.
- $H_7$ : Different interfaces influence adaptive behaviors.

In the rest of this section, we will describe the three web sites that we used as benchmarks in the user study. We then describe our methodology and setup.

### 5.1 Benchmarks

We developed three image-rich web sites for our user study: *Synthetic*, *Photo Album*, and *Poster Ads*. For each site, we gave users a specific set of tasks to perform. Users must improve image fidelity until enough detail is available to complete the task.

We asked users to perform specific tasks in order to simulate personal interests in the content and prevent random browsing behavior. In the real world, users browse the sites that they are interested

in and have specific motivations for looking at content (e.g., read the latest news, or buy a product with a given feature). We argue that it is this motivation that determines the way in which an object should be adapted. Unfortunately, such motivation cannot form spontaneously in a lab experiment and requires us to use specific tasks to compensate for this limitation.

We designed each of our three web sites with a specific purpose in mind. *Synthetic* represents the best case for Chameleon, where users use images for the same purpose and steps have been taken to reduce variation among subjects. *Photo Album* is a more realistic scenario, where users interact with real images downloaded from the web. These users still share the same task but there is more opportunity for variation among individuals. Finally, *Poster Ads* reflects the most challenging scenario, where users use the same images to perform different tasks.

#### 5.1.1 Synthetic

This is a controlled experiment where users are asked to recognize 6 images that contain words made up of 10 random letters in one of three font sizes (14, 20, and 26pt). The font size affects the fidelity level at which the word can be recognized (smaller fonts require higher fidelity levels). Users are presented one image at a time as shown in Figure 1. Together with the image, users are presented with 4 text buttons, and are asked to choose the button that corresponds to the image. Should the user select the wrong button, the system prompts the user to select again. However, the entry for this particular trial is discarded so that our analysis only considers the traces of users who correctly performed the task.

#### 5.1.2 Photo Album

This web site consists of two pages. Each page has 6 photographs arranged in a single column. On the first page, users are asked to determine how many of the 6 pictures show a popular movie actor. In the second page, users are asked to identify those pictures that show a widely-recognized politician. Since multiple images are displayed on a single page, users have the freedom to interact with images in different order, as opposed to the controlled

experiment where users had to interact with images in sequence. All of our test subjects were previously familiar with the appearance of the actor and the political personality used in this experiment.

The design of this experiment is similar to that of the synthetic experiment but since the task depends on a user’s capability for recognizing the individual in the photograph, we expect to see more variance in the adaptation requirements of different users for the same object.

### 5.1.3 Poster Ads

We designed this web site (and its related tasks) to explore a situation where the same content is used to accomplish multiple tasks. The web site contains 4 poster images for a large electronics store. On each image, there are multiple products listed with their promotion price. Users are asked to perform one of three tasks. The tasks involve browsing the posters in sequence, and obtaining relevant information, such as prices for on-sale items.

We expect that users performing the same task would have similar adaptation requirements but these would differ significantly from the adaptation requirements of users performing other tasks.

## 5.2 Methodology and Setup

We asked 30 University students to participate in our user study. Test subjects were asked to perform the Synthetic experiment with both the tapping and the multi-tab interfaces. A fully randomized, within-subjects design was used. The two interfaces (tapping and multi-tab) were counter balanced between subjects. That is, all 30 students performed the synthetic experiment on both interfaces; 15 users performed the experiment with the tapping interface first, while the other 15 used the multi-tab interface initially. In the Photo Album and Poster Ads experiment, all users performed the tasks only with the tapping interfaces.

In the Poster Ads experiments, we divided users into three equal-sized groups and assigned each group one of the three available tasks. Users were randomly assigned to groups at the start of the experiment.

We provided test subjects with instructions both on-line and on paper sheets. The users were told that they were accessing content over a CDMA 1X cellular connection, and that an adaptive proxy would transcode images into low fidelity to reduce download time. They were also given a brief training session at the beginning of the experiment to become familiar with the system.

We purposely forced the system to be in Learning Mode through all our experiments so that we could collect traces that would tell us the minimum fidelity that enabled our experiment participants to complete tasks. We designed our incentives structure to encourage users to finish the tasks as quickly as they could. This setup encourages users to stop refining image fidelity once image quality is sufficient for the task at hand.

Finally, to reduce variation due to differences in cellular coverage, we conducted all experiments in the same location during business hours. Even with these precautions, we experienced significant variations in CDMA 1X network bandwidth between test subjects.

## 6. EXPERIMENT EVALUATION

In this section, we start by providing an analysis of the traces collected during our user study. Next, we evaluate the performance of our prediction algorithms. Finally, we analyze the impact of different user interfaces on adaptive behavior.

## 6.1 Trace Analysis

On average, users were able to complete the user study with just a fraction of the bandwidth required by the full fidelity content (44% for Synthetic, 19% for Photo Album, and 49% for Poster Ads). Thus, we conclude that **(R1<sup>7</sup>) fidelity adaptation can significantly reduce bandwidth consumption.**

Figure 3 shows a histogram of the fulfillment fidelities for one representative image from each of our three web sites. The histogram of the Synthetic image shows strong locality in fulfillment fidelity with 80% of user study subjects selecting fidelity 3 for this image. The histogram also shows that even within a controlled experiment, there are small variations in fulfillment fidelity with some users selecting fidelity 2 and fidelity 4. We attribute this variation to differences in user perception. The histogram of the Photo Album image shows a similar pattern to the Synthetic image with 43% of subjects selecting fidelity 3 for this image; however, we see a higher variance in fulfillment fidelities. We attribute the higher variance to differences in the ability of users to recognize the person in the photographs. When we consider users who requested fidelity levels 2, 3 and 4, we find that 87% of users fall into this range. Based on the observation that the fulfillment fidelity of most users is clustered over a small range of fidelities, and the fact that all users performed the same tasks in the Synthetic and Photo Album experiments, we conclude that **(R2) users with the same usage semantics have similar adaptation requirements.**

Other Synthetic images show similar strong locality with the caveat that the most popular fulfillment fidelity changes based on the font size of the letters in the image. Overall the most common fulfillment fidelities were 3, 5, and 9 for images with large, medium and small text font size, respectively. Other images in the Photo Album experiment show similar clustering of fulfillment fidelity, but the location of the cluster differs across images. Thus, we conclude that **(R3) adaptation requirements vary from image to image.**

Finally, the histogram for the Poster Ads image shows a distinctive pattern where fulfillment fidelity is concentrated in three different clusters. This occurs because we gave users three distinct tasks to perform over the same set of images. Thus, we conclude that **(R4) users with different usage semantics have different requirements for adaptation.**

Based on observations R2, R3 and R4, we conclude that **(R5) usage semantics affect adaptation requirements.**

### 6.1.1 Discussion

The commonality in fulfillment fidelity across users observed in the histograms of Figure 3 motivates a history-based approach to fidelity prediction; however, the spread in fulfillment fidelity in the histograms (due to differences in user preferences or semantics of the object vis-a-vis the task) suggests that predictions based only on history (e.g., serve mean fidelity) are likely to result in significant user interaction. Therefore, to optimize fulfillment time, a predictive approach also needs to consider the latencies associated with user interaction. It is this realization that motivated the Chameleon PAS policy.

## 6.2 Prediction Performance

In this section, we evaluate the performance of Chameleon in prediction mode. We start by describing our methodology for calculating the performance of various policies. Next, we evaluate Chameleon when the goal is optimizing fulfillment time. Subsequently, we evaluate our system when the user wishes to trade bandwidth consumption for interactions. All of the results pre-

<sup>7</sup>The R in section 6 correspond to H in section 5.

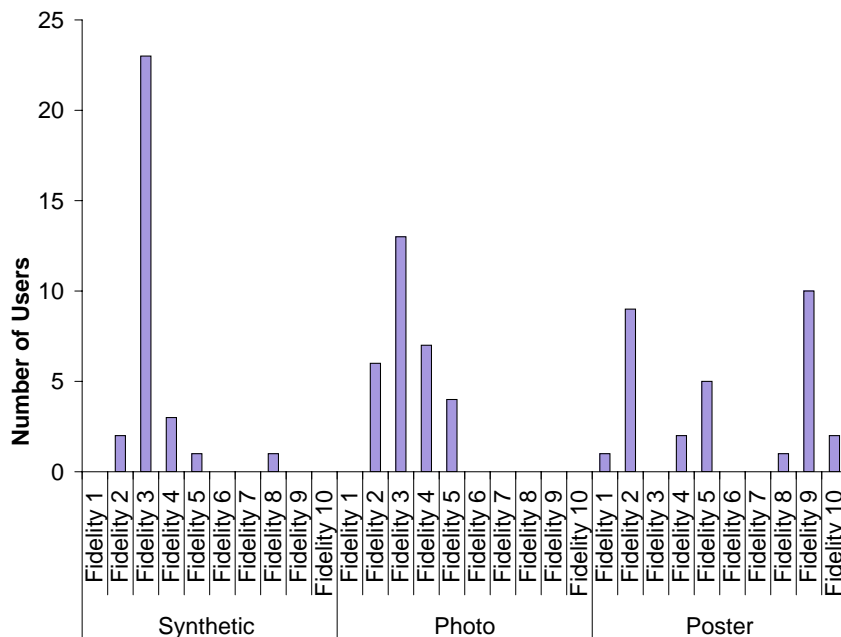


Figure 3: Fulfillment fidelity distribution for 3 web sites.

sented in this section are for the tapping interface. We discuss results for the multi-tab interface in section 6.3.

### 6.2.1 Methodology

We consider 4 adaptation policies in our evaluation: *Oracle*, *NoAdaptation*, *Mean* and *PAS*. *Oracle* illustrates a perfect predictive adaptation policy that loads images at the user’s fulfillment fidelities captured in the user study traces, and as such *Oracle* provides an upper bound to the performance of the adaptation system. In contrast, *NoAdaptation* corresponds to a simple policy that loads all images at their full fidelity. *NoAdaptation* provides a lower bound on the performance of the adaptation system below which adaptation becomes counter-productive. The fulfillment time for *Oracle* and *NoAdaptation* consists exclusively of the time taken to transfer image data over the network, as neither policies involve any user interaction. We estimate the fulfillment time of a given image for each user under *Oracle* and *NoAdaptation* by dividing the size of the image by the average network bandwidth and latency experienced by the user while performing the user study.

*Mean* is representative of a class of Chameleon policies that consider history but do not account for user interaction time. We also experimented with other statistics-based policies. While different policies perform best for individual web sites (e.g., *Mode* performs best for Photo Album), *Mean* was the best overall performer. We use the traces collected in our user study to estimate the expected performance of *Mean* in steady state. For each of the 30 users in the traces, we simulate steady state by assuming that the system has gathered history data by *previously* interacting with the other 29 users in learning mode. When *Mean* predicts a fidelity level that is equal or higher than the user’s fulfillment fidelity, fulfillment time consists only of the time required to download the predicted fidelity. However, when the prediction is lower than the user’s fulfillment fidelity, fulfillment time consists of the sum of the time required to download the fulfillment fidelity and the product of the user’s average interaction time and the number of interactions

needed to get from the predicted fidelity to the user’s fulfillment fidelity. For example, if a user’s fulfillment fidelity for an image is 7, but *Mean* predicts fidelity 5, then the fulfillment time is the time to download the image at fidelity 7 plus two times the user’s average interaction time.

We estimate the expected performance of *PAS* in steady state similar to *Mean*. For each of the 30 users, we compute their *PAS* schedule for each image based on the history of the other 29 users, the average network bandwidth experienced by the user, and the user’s average interaction time. To determine the user’s expected fulfillment time for a specific image, we first determine the lowest fidelity level in the user’s *PAS* adaptation schedule that is equal or higher than the user’s fulfillment fidelity captured in the trace. We refer to this fidelity as the *target* fidelity. We then compute fulfillment time by adding the time it takes to download the target fidelity under the average network bandwidth experienced by the user to the product of the user’s average interaction time and the number of interactions needed to get to the target fidelity. For example, if the *PAS* schedule of a user for an image is {2,5,7,10} and the user’s fulfillment fidelity for the image is 6, then the expected fulfillment time equals the time to download the image at fidelity 7 plus two times the user’s average interaction time.

Overall, the average user interaction time for subjects in our user study was 2388ms, with a standard deviation of 1421ms and the average CDMA 1X network bandwidth was 83 Kbps with a standard deviation of 16.9 Kbps.

### 6.2.2 Optimizing Fulfillment Time

Figures 4 and 5 show the average per-page fulfillment time and bandwidth consumption across users for *Oracle*, *NoAdaptation*, *Mean*, and *PAS* over CDMA 1X<sup>8</sup>. Error bars in all figures show

<sup>8</sup>The fulfillment time and bandwidth consumption for Photo Album are significantly higher than that of Synthetic and Poster Ads because web pages in Photo Album each have 6 images while those in Synthetic and Poster Ads only contain single images.

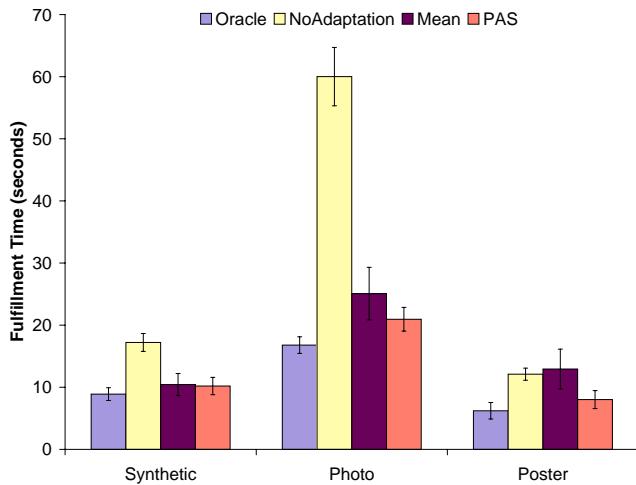


Figure 4: Fulfillment time over CDMA 1X.

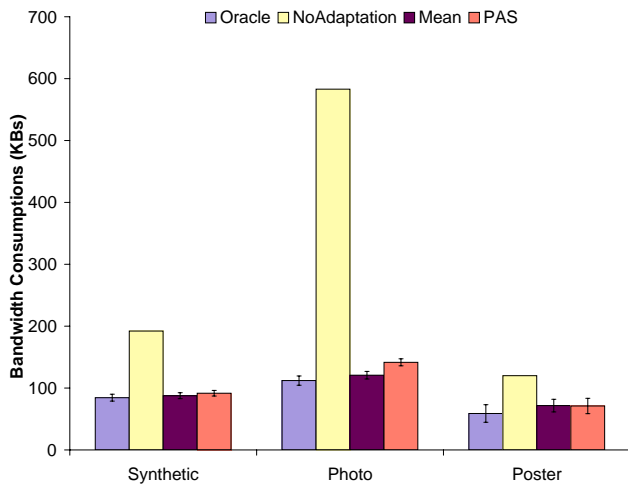


Figure 5: Bandwidth consumption over CDMA 1X.

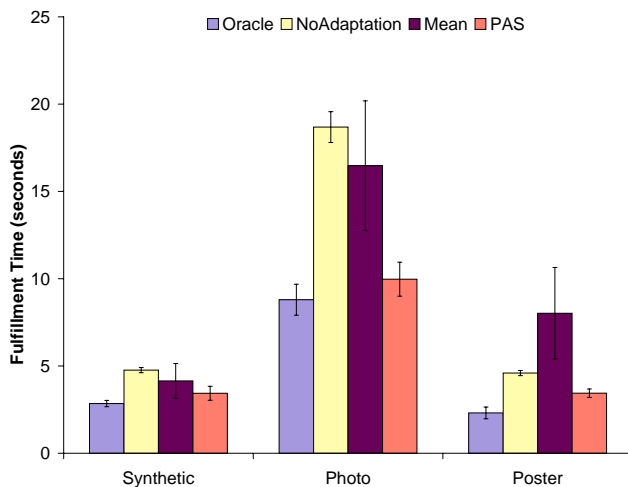


Figure 6: Fulfillment time over simulated 3G.

95% confidence intervals<sup>9</sup>.

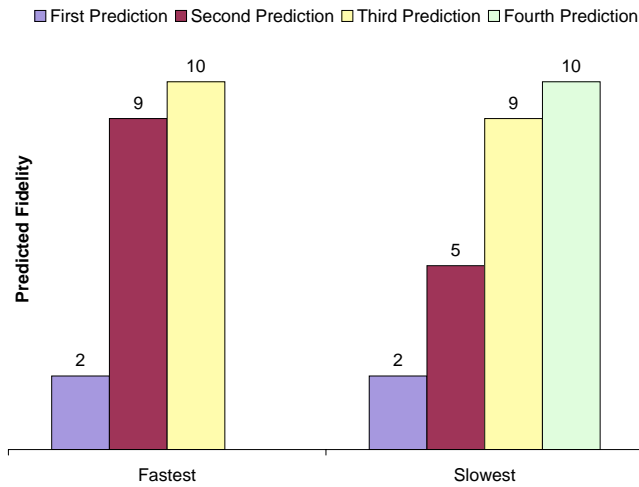
We see that both *Mean* and *PAS* achieve large reductions in fulfillment time and bandwidth consumption compared to *NoAdaptation* for the Synthetic and Photo Album web sites. Also, in comparison to *Oracle*, both policies come within 13% in terms of fulfillment time and 4% in terms of bandwidth consumption, suggesting that both policies are very effective and that there is little room for improvement. For the Poster Ads web site, however, although *PAS* reduces fulfillment time by 33% and gets within 14% of *Oracle*, *Mean* increases fulfillment time by 6%. It should be noted that both *Mean* and *PAS* consume similar amounts of bandwidth, which is significantly lower than that consumed by *NoAdaptation*. The difference in fulfillment time is attributable to the amount of interaction each policy requires. *PAS* correctly identifies the different fidelity levels that are suitable for the varying tasks being carried out on the Poster Ads site, and therefore quickly jumps to the next relevant fidelity once the user asks for an improvement. In contrast, *Mean* requires the user to interact several times when it mispredicts.

Figure 6 shows the average per-page fulfillment time across users for *Oracle*, *NoAdaptation*, *Mean*, and *PAS* over a simulated 3G link with an effective per user bandwidth of 384 Kbps<sup>10</sup>. The figure shows that even for the much higher bandwidth of 384 kbps (three times higher than the data rate of wide area mobile data services currently available in North America), adaptation is still highly beneficial. Overall, *PAS* still achieves savings for all three web sites (e.g., for Photo Album, *PAS* provides a 47% reduction and comes within 6% of *Oracle*). These results show that *PAS* is able to adapt its prediction schedules and take advantage of the increase in available bandwidth. *PAS* increases its bandwidth consumption by 20% compared to the CDMA 1X results, but the download time is more than offset by a reduction in user interaction time. In contrast, *Mean*'s predictions remain fixed and its performance suffers greatly as the relative contribution of user interaction time to fulfillment time increases. Note that even though the absolute fulfillment time savings for adaptation may become smaller in such high bandwidth environments, users still benefit from conserving significant amounts of bandwidth.

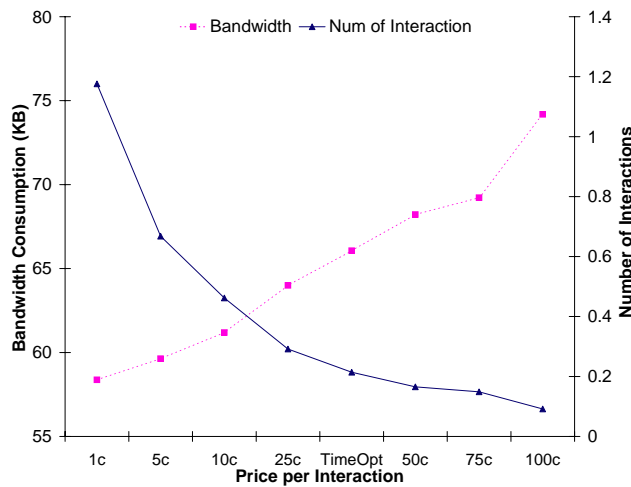
Figure 7 illustrates how *PAS* generates personalized adaptation schedules that (in addition to history) take into account the prevailing bandwidth and latency conditions, as well as the user's average interaction time. Figure 7 shows two adaptation schedules for an image in the Poster Ads web site. The left and right schedules correspond to the users that experienced the fastest (107 Kbps) and slowest (54 Kbps) average network bandwidth, respectively. Bars in the schedule represent the order in which image fidelity will be improved. For example, for the user with slow connectivity, *PAS* provides the image initially at fidelity 2, and at fidelities 5, 9 and 10 upon user request. As expected, *PAS* generates a schedule with more predictions for the user with slower connectivity, and a schedule with fewer predictions for the user with the faster connection speed. In general, when the average user interaction time is high, or the connection speed is relatively fast, *PAS* serves users at a higher initial fidelity level and improves object fidelity in greater strides. It is worth noting that the schedule for the user with slow connectivity corresponds to the fidelity levels that most users will select in this image. This suggests that *PAS* is capable of identifying user selection "hot spots" on-the-fly.

<sup>9</sup>In Figure 5, the *NoAdaptation* policy does not have error bars since every user is served with all of the content at full fidelity.

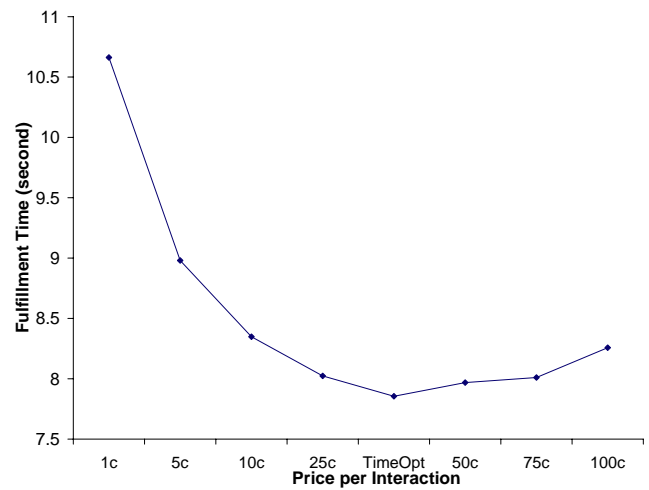
<sup>10</sup>While the nominal bandwidth for a 3G cell is rated at around 2 Mbps, effective per-user bandwidth is expected to be 384 Kbps or lower.



**Figure 7: Personalized prediction schedules for users with fastest and slowest average network bandwidth.**



**Figure 8: Tradeoff between user interaction and bandwidth consumption.**



**Figure 9: Change to fulfillment time for different bandwidth/interaction tradeoffs.**

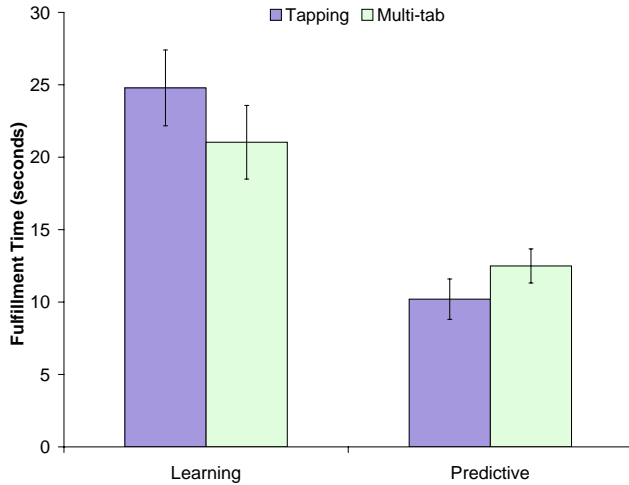
### 6.2.2.1 Discussion.

These results show that the performance (with respect to fulfillment time) of statistical policies that do not take into account user interaction time can vary erratically across web sites and bandwidth conditions. Thus **(R6) Prediction based purely on history will reduce bandwidth consumption, but may not always result in optimal fulfillment time.** In contrast, *PAS* proves robust across web sites and evaluation conditions. Moreover, the tightness of the confidence interval bars shows that *PAS* provides stable performance across users. By taking into account the relative cost of network transfer and user interaction, *PAS* seamlessly adjusts its prediction schedule to account for differences in object size across web sites, as well as changes in bandwidth conditions. Finally, *PAS* is effective even when content is used for multiple tasks with different adaptation requirements.

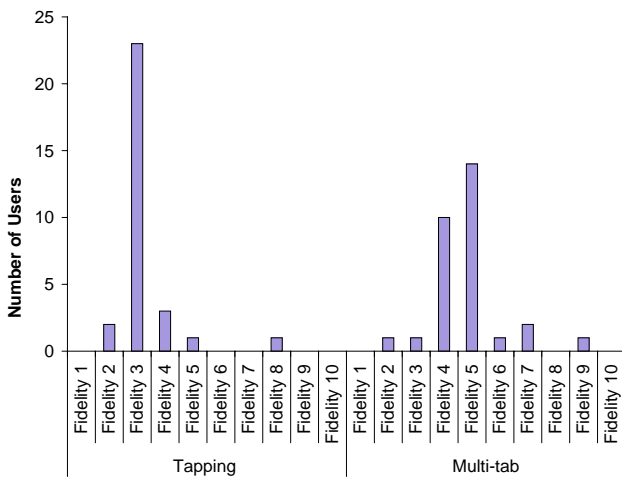
### 6.2.3 Trading Bandwidth for User Interaction

Figures 8 and 9 show the effect of varying the monetary value associated with user interaction on bandwidth consumption, average number of user interactions and fulfillment time, averaged across all images in our user study for a hypothetical case where users pay \$10 per MB of data downloaded (or 1 cent per KB)<sup>11</sup>. The X-axis in Figure 8 represents different unit prices for interactions. The Y-axis on the left shows the average bandwidth consumption per image. The Y-axis on the right shows the average number of interactions per image. Clearly, as interaction becomes cheaper, the user will be required to provide more interactions, with a resultant decrease in average bandwidth consumption. In contrast, as interaction becomes more expensive, the system will require the user to perform fewer interactions, resulting in higher bandwidth consumption. Figure 8 shows that for the 3 web sites used in our user study, an average user can reduce her bandwidth consumption by close to 20 KB (save 20 cents) per image if she is willing to interact with the system an average of 1.2 times per image. Figure 9 shows how fulfillment time varies from the optimal for different bandwidth/interaction tradeoffs.

<sup>11</sup>In practice, it is not uncommon for users to pay as much as \$30 per MB.



**Figure 10: Average fulfillment time in learning and prediction mode with tapping and multi-tab interface for Synthetic web site.**



**Figure 11: Fulfillment fidelity distribution of tapping and multi-tab interface for a single image**

### 6.3 Effects of User Interface

Figure 10 shows average fulfillment time for the Synthetic web site for Chameleon in learning and prediction modes with the *tapping* and *multi-tab* interfaces (refer to Section 3.1 for details on the two interfaces). In prediction mode, Chameleon uses *PAS* to optimize fulfillment time. This figure illustrates an interesting tradeoff. We see that multi-tab achieves lower fulfillment times in learning mode (T-test  $p = 0.006$ ), but results in higher fulfillment times once Chameleon switches to prediction mode. While multi-tab requires fewer user interactions in learning mode (T-test  $p < 0.0001$ ), it also encourages users to select higher fulfillment fidelities (T-test  $p < 0.0001$ ), which in turn results in higher bandwidth consumption and longer download times. Moreover, as Figure 11 shows multi-tab results in a wider spread in the range of fulfillment fidelities, which degrades the quality of predictions. Therefore, there is a tradeoff between the effort that users need to invest in training the system and the potential benefits once the system moves into prediction mode. Since we expect that Chameleon will be in prediction mode most of the time, tapping appears to be a better choice of interface. Thus we conclude that **(R7) different interfaces impact adaptive behaviors.**

## 7. RELATED WORK

Numerous research efforts have considered content adaptation for mobile devices [5, 9, 12, 15, 17, 22, 24, 26, 28, 29, 31, 32, 37, 38], and there has even been deployment of a few commercial adaptation systems [5, 21]. In general, the goal of adaptation is to tailor content so that it becomes more suitable for users than its original form. Manual adaptation techniques, such as WAP [41], strive to provide fine-grain content customization. However, these techniques place significant onus on content creators who are required to maintain multiple versions of their content to support a plethora of devices. As a result, deployment has been limited to a small set of high-traffic Web sites that can afford the high cost of hand-tailoring content for mobile clients; and even then, support is limited to a few popular devices and content is updated at a lower frequency than the main site. For example, CNN currently updates its mobile version only twice a day; in contrast, its main page is updated several times per hour.

Automatic adaptation systems [5, 12, 15, 26, 29, 38, 39] that transform content on-the-fly are a promising alternative. The main challenge for automatic content adaptation is the design of effective adaptation policies. The two main techniques for policy generation are rule-based [5, 19, 37, 38, 39] and constraint-based [13, 26, 28, 39] adaptation. In both approaches, policies are defined using high-level programming languages or mathematical formulas. Unfortunately, content providers cannot be expected to provide constraints or rules for every data object, as this would not be very different from supplying customized content for every client type. As a result, small sets of rules apply to broad sets of content (e.g., all JPEG images are adapted the same way independent of their purpose or value to the user). In contrast, URICA takes into account usage semantics of content.

URICA builds on our preliminary work on history-based content adaptation [27], where we presented a limited evaluation of a proof-of-concept fidelity adaptation prototype that optimized bandwidth consumption. In this paper, we show that focusing solely on bandwidth optimization can lead to browsing latencies that are higher than when no adaptation is performed. The Chameleon system that we present in this paper allows users to minimize their overall latency for browsing adapted content or achieve their desired tradeoff between bandwidth consumption and the need to in-

teract with the system.

Contemporary real-time multimedia systems often utilize adaptive techniques alongside traditional scheduling and admission control mechanisms in order to meet Quality of Service (QoS) requirements [1, 6, 20, 25]. Factors such as server overload or congestion within the network can trigger degradation in the quality of the multimedia content that is provided to the client while meeting real-time constraints.

Narayanan [28] used history logs of past system performance to predict the operating mode of an application that achieves a specified battery life and maximizes content fidelity. In URICA, the history that is used consists of past user preferences for the adaptation decision. Also, in Narayanan's approach the adaptation decision does not consider content usage semantics, which is the case in URICA.

URICA is related to efforts on recommendation-based systems [2, 3, 10, 18, 30, 40], in which people collaborate to help one another perform filtering by recording their reactions to content they access. URICA is a radical new use of the history-based recommendation concept – adaptation prediction. Whereas previous efforts have focused on predicting what content to provide, URICA focuses on the question of how to adapt this content for display on mobile devices.

## 8. CONCLUSIONS

We presented Usage-awaRe Interactive Content Adaptation (URICA), a technique for automatic content adaptation that makes its adaptation decision based on usage semantics. URICA allows users to change how content is adapted whenever it is unsuitable for their purposes and utilizes this feedback to improve future adaptation decisions. To evaluate our technique, we developed Chameleon, a system that performs fidelity adaptation of images and allows users to conserve bandwidth as they browse the web over cellular links. We conducted a user study in which participants used Chameleon to browse image-rich web pages in order to complete tasks that simulated various usage semantics. The user study showed that usage semantics impact adaptation requirements. Chameleon reduces the latency of browsing content by up to 65% and can reduce bandwidth consumption by up to 80%. Users can also utilize Chameleon to exchange bandwidth consumption, which for many cellular users has a direct monetary cost, for user involvement in the adaptation process. Chameleon is robust in that it works well under different network conditions and achieves large savings even when the same content has multiple usage semantics. Finally, we showed that the choice of user interface for providing feedback can influence adaptive behavior.

In the future, we plan to extend Chameleon for energy adaptation. We also plan to explore the effect of other factors such as context on the user's adaptation requirements.

## Authors' contributions

The first two authors contributed equally to the paper.

## REFERENCES

- [1] T. F. Abdelzaher and K. G. Shin. Qos provisioning with qContracts in web and multimedia servers. In *IEEE Real-Time Systems Symposium*, pages 44–53, 1999.
- [2] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [3] M. Balabanovic, Y. Shoham, and Y. Yun. An adaptive agent for automated web browsing. *Journal of Visual Communication and Image Representation*, 6(4), 1995.
- [4] D. Berlind. Which network—cdma or gprs? no easy answers. ZDNet, Jan. 2003. <http://techupdate.zdnet.com>.
- [5] K. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey. Transcoding: Extending e-business to new environments. *IBM Systems Journal*, 40(1):153–178, 2001.
- [6] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A distributed real-time MPEG video audio player. In *Network and Operating System Support for Digital Audio and Video*, pages 142–153, 1995.
- [7] R. Chakravorty, S. Katti, I. Pratt, and J. Crowcroft. Using top flow-aggregation to enhance data experience of cellular wireless users. *IEEE Journal of Selected Areas of Communications*, 23(6), June 2005.
- [8] M. C. Chan and R. Ramjee. Tcp/ip performance over 3g wireless links with rate and delay variation. In *International Conference on Mobile Computing and Networking (MobiCom)*, Atlanta, GA, Sept. 2002.
- [9] S. Chandra, A. Gehani, C. S. Ellis, and A. Vahdat. Transcoding characteristics of web images. In *Proceedings of the 2001 Multimedia Computing and Networking Conference (MMCN'01)*, San Jose, California, Jan. 2001.
- [10] CiteSeer. <http://citeseer.ist.psu.edu/>.
- [11] E. de Lara, R. Kumar, D. S. Wallach, and W. Zwaenepoel. Collaboration and multimedia authoring on mobile devices. In *International Conference on Mobile Systems, Applications, and Services*, San Francisco, California, May 2003.
- [12] E. de Lara, D. S. Wallach, and W. Zwaenepoel. Puppeteer: Component-based adaptation for mobile computing. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, San Francisco, California, Mar. 2001.
- [13] Y. Dotsenko, E. de Lara, D. S. Wallach, and W. Zwaenepoel. Extensible adaptation via constraint solving. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, Callicoon, New York, June 2002.
- [14] J. Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Transactions on Computer Systems (TOCS)*, 22(2), May 2004.
- [15] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. *SIGPLAN Notices*, 31(9):160–170, Sept. 1996.
- [16] A. Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. *IEEE Personal Communications*, 5(4):10–19, 1998.
- [17] A. Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. *IEEE Personal Communications*, 5(4):10–19, Aug. 1998.
- [18] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [19] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. Dynamic adaptation in an image transcoding proxy for mobile web browsing. *IEEE Personal Communications*, 5(6):8–17, 1998.
- [20] Q. He and K. Schwan. Iq-rudp: Coordinating application adaptation with network transport, 2002.
- [21] iAnywhere Solutions. Avantgo. [www.avantgo.com](http://www.avantgo.com).

- [22] R. H. Katz. Adaptation and mobility in wireless information systems. *IEEE Personal Communications*, 1(1):6–17, 1994.
- [23] T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1), Jan. 2002.
- [24] T. Kunz, M. E. Shentenawy, A. Gaddah, and R. H. Hafez. Image transcoding for wireless WWW access: the user perspective. In *the SPIE Multimedia Computing and Networking (MMCN2002)*, San Jose, CA, Jan. 2002.
- [25] K. Li, J. Walpole, D. McNamee, C. Pu, and D. C. Steere. A rate-matching packet scheduler for real-rate applications. In *Proceedings of Multimedia Computing and Networking 2001*, San Jose, California, Jan. 2001.
- [26] W. Y. Lum and F. C. Lau. A context-aware decision engine for content adaptation. *IEEE Pervasive Computing*, 1(3):41–49, July 2002.
- [27] I. Mohomed, A. Chin, J. Cai, and E. de Lara. Community-driven adaptation: Automatic content adaptation in pervasive environments. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA '04)*, pages 124–133, Lake District National Park, UK, Dec. 2004. IEEE Computer Society.
- [28] D. Narayanan, J. Flinn, and M. Satyanarayanan. Using history to improve mobile application adaptation. In *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, Monterey, California, Dec. 2000.
- [29] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Agile application-aware adaptation for mobility. *Operating Systems Review (ACM)*, 51(5):276–287, Dec. 1997.
- [30] N. Ohsugi, A. Monden, and K. Matsumoto. A recommendation system for software function discovery. In *Proceedings of the 9th Asia-Pacific Software Engineering Conference (APSEC2002)*, Gold Coast, Queensland, Australia, Dec. 2002.
- [31] T. Phan, G. Zorpas, and R. Bagrodia. Middleware support for reconciling client updates and data transcoding. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, June 2004.
- [32] C. Poellabauer and K. Schwan. Energy-aware media transcoding in wireless systems, 2004.
- [33] A. Qureshi and J. Guttag. Horde: Separating network striping policy from mechanism. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Seattle, WA, June 2005.
- [34] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MAR: A commuter router infrastructure for the mobile internet. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, June 2004.
- [35] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Fifteenth ACM Symposium on Principles of Distributed Computing*, Philadelphia, Pennsylvania, May 1996.
- [36] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 2001.
- [37] B. N. Schilit, J. Trevor, D. M. Hilbert, and T. K. Koh. Web interaction using very small internet devices. *IEEE Computer*, 35(10):37–45, 2002.
- [38] J. R. Smith, R. Mohan, and C.-S. Li. Content-based transcoding of images in the Internet. In *Proceedings of the IEEE International Conference on Image Processing*, Chicago, Illinois, Oct. 1998.
- [39] J. R. Smith, R. Mohan, and C.-S. Li. Transcoding internet content for heterogeneous client devices. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, Monterey, California, May 1998.
- [40] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: a system for sharing recommendations. *Commun. ACM*, 40(3):59–62, 1997.
- [41] WAP Forum. Wireless application protocol architecture specification, Apr. 1998. Available at: <http://www.wapforum.org/what/technical/arch-30-apr-98.pdf>.