Context-Aware Interactive Content Adaptation

Iqbal Mohomed, Jim Chengming Cai, Sina Chavoshi, Eyal de Lara
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada
(iq, jcai, delara)@cs.toronto.edu and sina.chavoshi@utoronto.ca

ABSTRACT

Automatic adaptation of content for mobile devices is a challenging problem because optimal adaptation often depends on the usage semantics of content, as well as the context of users (e.g., screen size of device being used, network connectivity, location, etc.). Usage-awaRe Interactive Content Adaptation (URICA) is an automatic technique that adapts content for mobile devices based on usage semantics. URICA allows a user who is unsatisfied with the system's current adaptation prediction to take control of the adaptation process and make changes until the content is suitably adapted for her purposes. The adaptation system learns from the user's modifications and adjusts its prediction for future accesses by other users.

This paper shows that it is possible to exploit user interaction to learn how to adapt content based on context. We introduce Feedback-driven Context Selection (FCS), an automatic technique that leverages user interaction to identify the context that has the most impact on adaptation requirements. We added context-awareness to URICA so that it makes adaptation predictions for a user based only on the history of the community of users that share the context identified by FCS. The result is an automatic adaptation system that provides fine grain adaptations that reflect both the user's context and the content's usage semantics. This level of fine grain adaptation was previously available only in content that was customized manually. Experiments with two context-aware URICA prototypes show that FCS correctly identifies the contextual characteristics that impact adaptation requirements, and that grouping users into communities based on context improves the performance of the adaptation system by up to 79%.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation, Human Factors

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.
MobiSys'06, June 19–22, 2006, Uppsala, Sweden.
Copyright 2006 ACM 1-59593-195-3/06/0006 ...$5.00.

Keywords

Content Adaptation, Context, Mobile Devices, Customization

1. INTRODUCTION

The need to adapt content for use on mobile devices has been long recognized [19, 20, 32, 33], and both manual and automatic approaches to content adaptation have been proposed. Manual adaptation techniques, such as WAP [38], have high cost for data producers who are required to maintain multiple copies of their content. Automatic content adaptation [3, 8, 12, 16, 22, 23, 28, 29, 30, 34, 35] is challenging because optimal adaptation often depends on the content's usage semantics and the user's context. For example, the optimal dimensions for rendering an image may depend on both the role the image plays as part of the user's task as well as her context, such as the screen size of the mobile device being used and the network bandwidth that is available. As such, an ideal adaptation system should customize content based on both usage semantics and context.

We have previously introduced the idea of utilizing user interaction to learn how to adapt content based on usage semantics [26, 27]. We developed Usage-awaRe Interactive Content Adaptation (URICA), an automatic technique that leverages user interaction to learn how to adapt content for mobile devices. URICA allows users who are unsatisfied with the system's current adaptation prediction 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 becomes part of the history that is used in making future adaptation predictions for other users. For example, a URICA system that adapts page layout for display on small screens allows unsatisfied users to resize individual images. The adaptation system learns from the user's modifications and adjusts its predictions for future accesses to the same images. For users that have the same context and use content for the same purpose, URICA achieves near-optimal adaptation performance. When content is being used for multiple purposes or when context differs across users, URICA continues to achieve benefits over no adaptation; however, optimal performance is not attained.

This paper shows that it is possible to exploit user interaction to learn how to adapt content based on context. This is a challenging problem because there is a large amount of context that can potentially affect adaptation requirements, and the context that is relevant can vary between content. For example, context such as the user's current location may be important in content that includes a map but otherwise have little bearing on adaptation requirements. We introduce an automatic technique called Feedback-driven Context Selection (FCS), which leverages user interaction obtained naturally during URICA's operation, to identify the context that has the most impact on content adaptation requirements.

We added context-awareness to URICA by using the contextual characteristics identified by FCS as the basis for grouping users into communities, and making adaptation predictions for users using only the restricted history of their communities. The result is an automatic adaptation system that provides fine grain adaptation, reflecting both the user's context and the content's usage semantics. This level of fine grain adaptation was previously available only in content that was customized manually.

We developed two context-aware URICA prototypes to evaluate the benefits of grouping users into communities based on contextual characteristics. The first prototype supports mobile devices with limited screen real-estate by adapting the layout and display size of images embedded within HTML pages. The second prototype provides image fidelity adaptation, which allows users to save bandwidth as they browse the web. We conducted controlled user studies where we asked participants to use our prototype adaptation systems to complete a fixed set of tasks as we varied the underlying context. Specifically, we experimented with four different contextual characteristics: device type, screen size, network bandwidth, and user location. Our results show that FCS correctly identifies the contextual characteristics that have the most impact on the adaptation requirements of an object, and that grouping users into communities based on these contextual characteristics improves the quality of the adaptation system's predictions by up to 79%.

This paper makes four contributions: (i) it shows that differences in context can affect content adaptation requirements; (ii) it shows that not all context information is equally important in determining adaptation requirements; (iii) it shows that is it possible to take advantage of user interaction to determine the contextual characteristics that have the most impact on the adaptation requirements of an object, and describes an automatic technique for determining the set of influential contextual characteristics; and (iv) it shows that grouping users into communities based on the set of influential contextual characteristics significantly improves the quality of URICA's adaptation predictions.

The rest of this paper is organized as follows. Section 2 provides an overview of URICA and our extensions to enable context-awareness. Section 3 presents the FCS technique, and describes how we augment URICA to group users into communities based on contextual characteristics. Section 4 details the implementation of two context-aware URICA prototype adaptation systems. Sections 5 and 6 describe the experiments we conducted with these prototypes and present the results of our evaluation, respectively. Section 7 describes related work. Finally, Section 8 concludes the paper and describes avenues for future work.

2. CONTEXT-AWARE URICA

Usage-awaRe Interactive Content Adaptation (URICA) [26, 27] is an automatic technique that adapts content for mobile devices based on usage semantics. In this section, we provide an overview of URICA, and detail the additional support required to add context-awareness to the system.

URICA leverages user interaction to learn how to adapt content. When serving content for the first time, URICA makes a default decision on how to adapt the content. Next, 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. For example, a user may choose to reorganize the layout of an HTML page to improve readability, or ask the system to improve the resolution of a specific image. The successful adaptation is recorded and becomes part of the history that is used in making future adaptation decisions for other users.

There are three components in a URICA system: an *adaptation proxy*, *adaptation clients*, and *content servers*. The adaptation proxy is at the heart of the URICA model; it mediates all accesses to content and determines how content should be adapted. The proxy contains a rich set of transcoders used to adapt content on-the-fly. Adaptation clients are mobile devices that run adaptive applications. These applications allow users to apply corrective measures that change the system's adaptation decisions. Depending on the type of adaptations being supported by a specific URICA implementation, the adaptive application may provide users with a variety of corrective measures (e.g., resize images, change video frame rate). The user can repeatedly apply corrections until she is satisfied with the adapted version of the content being presented to her. Content servers are arbitrary data repositories, such as web servers, that store the content that the user wishes to access, and are unaware of the adaptation process.

To support context-awareness, we augment the functionality of adaptation clients and the adaptation proxy in the URICA model. For the remainder of the paper, we will refer to the augmented system as Context-Aware URICA (CA-URICA). In order to sense the user's context, adaptation clients can include sensor modules, which may be hardware or software components that provide context information. Context information includes physical device characteristics, such as device type and screen size, as well as more dynamic characteristics, such as user location, battery power level, and available storage and bandwidth. Context information is communicated to the adaptation proxy together with user requests for objects and refinements. It should be noted that while context information can be explicitly provided by a sensor, client software or the user, it may be possible for the adaptation proxy to automatically infer some context, such as the user's network bandwidth. The adaptation proxy utilizes our Feedback-driven Context Selection (FCS) technique to determine relevant context, groups users into communities based on the identified context, and maintains a history of satisfactory adaptations for every community. The FCS technique and grouping mechanism will be described in detail in the next section. In the remainder of this section, we describe how the adaptation proxy makes adaptation decisions.

When a user requests an object, the proxy uses the context information bundled with the request to determine the community that the user belongs to, and makes an adaptation decision based on the history of successful adaptations of this community. We assume that for every type of adaptation, it is possible to enumerate all of the adaptations that can be performed. As such, for every object being adapted (e.g., a specific image in a web page), the proxy keeps track of how often each of the possible adaptations are deemed satisfactory by members of a community. For example, Figure 1 shows a histogram of the image sizes of a specific image that have satisfied past users of a given community (assuming the image is available at 10 different sizes), and the prediction generated by the Mean policy, which serves the average of the sizes that satisfied past users. The semantics of the history captured in the histogram and the choice of the policy that is used to make the adaptation decision depend on the type of adaptation being performed. For example, for adaptations that have a natural ordering (e.g., fidelity or display size adaptation of images), serving the average value that satisfied past users is often reasonable. In contrast, for adaptations where no natural ordering exists (e.g., layout adaptation of images on a web page), it may be reasonable to pick the adaptation that satisfied most users in the past.

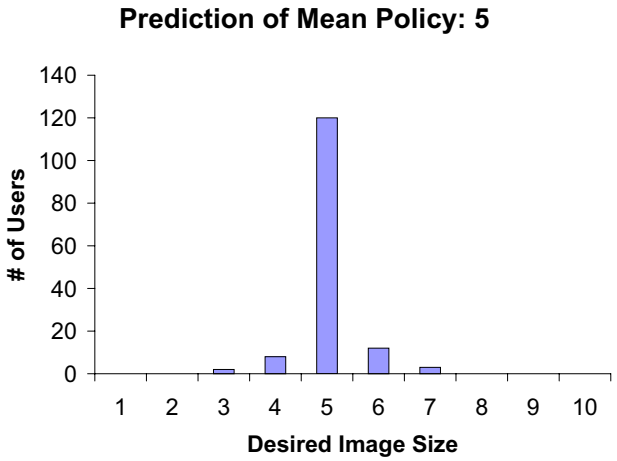

Figure 1: Histogram of image display sizes that satisfied past users. The vertical axis shows the number of users who desired each of the adaptations on the horizontal axis.

3. CONTEXT-BASED GROUPING

In this section, we describe how CA-URICA uses context information to group users into communities. We start by motivating why grouping users based on all available context is not feasible. Next, we present our Feedback-driven Context Selection (FCS) technique for automatically identifying context that has a significant impact on adaptation requirements, and describe how it is used by CA-URICA to group users into communities. Finally, we discuss deployment issues.

3.1 Motivation

Grouping users into communities based on context is a challenging problem because there are a large number of contextual characteristics that can potentially affect adaptation requirements (e.g., device type, screen size, network bandwidth and latency, cost per byte[1] , user location, time of day, browsing history, preferences, and battery level), with several possible values for each contextual characteristic (e.g., device type may be defined to be one of SmartPhone, PDA, In-Vehicle Terminal, and Laptop). Thus, a naive approach that groups users into communities based on all possible combinations of user context is not practical as the number of possible communities grows exponentially in the number of contextual characteristics. Moreover, the system would have to encounter at least some users in each community before the quality of predictions for that community converges. We would likely have very small communities where even a few outliers can inversely impact prediction quality. On the other hand, a static approach that uses a small number of fixed communities for all content is likely to be ineffective as the set of contextual characteristics that are significant may vary based on the content being adapted, as well as the type of adaptation being performed.

To address these challenges, we devised Feedback-driven Context Selection (FCS), a technique that automatically identifies the contextual characteristics that have a significant influence on adaptation requirements. These influential contextual characteristics are used by CA-URICA to group users into communities.

[1]Most cellular data plans have limited bandwidth allowances and charge users for additional data downloads.

3.2 Feedback-driven Context Selection

For any given object, we expect that there is significant diversity in the context of users who are accessing it. Feedback-driven Context Selection (FCS) is an automatic technique that dynamically identifies the contextual characteristics that have a significant influence on adaptation requirements using the user feedback that is naturally received in the course of CA-URICA's operation. The premise behind FCS is that we can consider the users encountered in the past as representatives of the overall user population. At the outset, FCS requires that the adaptation proxy be configured with a list of contextual characteristics, which are candidates for consideration. During operation, the adaptation proxy starts by associating each object with a single community. Even though the user's context does not influence the adaptation prediction in this situation, the proxy keeps track of context of past users along with the adaptation history. Once sufficient history has been accumulated for a particular contextual characteristic (e.g., device type), the adaptation proxy performs a profiling experiment to determine whether the previously encountered users would have experienced a substantial improvement in performance had they been partitioned into distinct communities based on the contextual characteristic under consideration. If so, the contextual characteristic is identified to be significant, and is used to partition users into separate communities.

We start by detailing the specification of candidate contextual characteristics to the adaptation proxy. Next, we describe the operation of the profiling experiment. Finally, we discuss storage considerations for the adaptation proxy.

3.2.1 Contextual Characteristics

The adaptation proxy can be configured to evaluate a wide range of candidate contextual characteristics. We assume that every contextual characteristic is a finite enumeration of values; each value is referred to as a context attribute. For example, device type is a contextual characteristic whose values could include the following context attributes: SmartPhone, PDA, In-vehicle Terminal, and Laptop. Real-valued contextual characteristics, such as compass heading or network bandwidth, must be discretized. This can be accomplished by partitioning the range of values into a finite number of intervals. For example, the user's current direction of movement can be discretized into 8 values: North, North East, East, South East, South, South West, West, and North West. When differentiating among users based on a particular contextual characteristic, we group together users with the same context attribute. For example, when differentiating based on device type, all SmartPhone users would be grouped together into a single community, and users with other device types would be grouped into their own device-specific community. Further, for every contextual characteristic, a user must have one and only one context attribute. Thus, each contextual characteristic can be used to completely partition the entire population. We expect that general candidate contextual characteristics that are likely to affect a large set of web content, such as screen size, device type, and network bandwidth, will be specified by the CA-URICA system operator. The adaptation proxy also lets content providers specify (if they choose to do so) additional application or domain-specific candidate contextual characteristics that are meaningful within the scope of specific web applications or domains. For example, a web site that includes a map may specify the user's current location and direction as contextual characteristics.

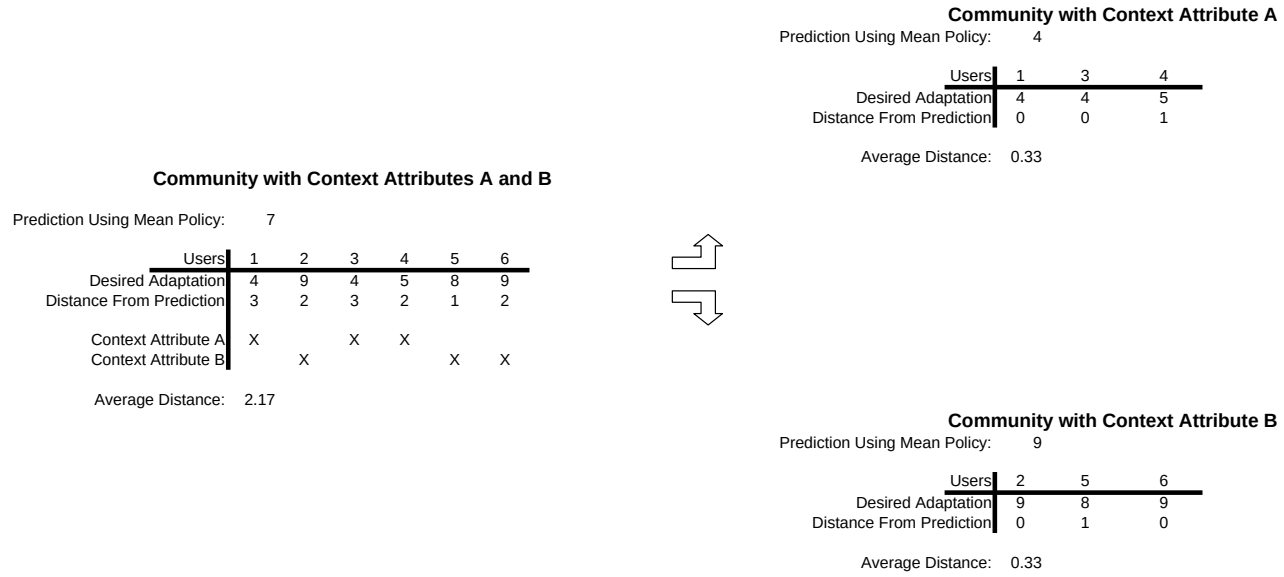

Figure 2: Simple example of profiling experiment.

3.2.2 Profiling Experiment

Our approach works as follows: an object (such as a video clip or an image) is initially associated with a single community. That is, regardless of their context, users requesting the object are trivially classified into this default community. For each context attribute, the system separately tracks the adaptation requirements of users having that context attribute. Once we have encountered a sufficient number of users having a particular contextual characteristic, we perform a profiling experiment in which we test if an improvement in performance can be attained by splitting users into multiple communities based on this contextual characteristic. If so, the community is split by this context. Otherwise, the existing community is maintained. The number of users that must be encountered before we conduct the profiling experiment is specified by T, which is a parameter provided by the operator of the adaptation system.

In the profiling experiment, the system compares the performance that users experienced when grouped with the existing community to that which would have been experienced if predictions had been made on the restricted history of only those users in the community that have the same context attribute[2]. The performance in both cases can be judged by a number of metrics such as distance from optimal prediction, number of mispredictions, bandwidth usage and energy consumption. This metric, which captures the quality of the adaptation system's predictions, is specified by the system operator and is selected based on the goal of the particular adaptation system. The primary metric used in our evaluation is *distance*, which is how far a prediction falls from the user's desired adaptation. The distance metric is simply the Euclidean distance between the system's prediction and the adaptation level desired by a user, and is applicable when there is a natural ordering in the adaptations that are available. Alternatively, a more complex metric, such as one that applies different weightings when a prediction is below a user's desired adaptation level than when it is above, can also be used. To determine whether splitting an existing community based on some contextual characteristic is worthwhile, we consider the improvement in the selected metric that is experienced by users in each of the prospective communities, along with the proportion of users that fall into that community. Suppose we are considering whether to split an existing community C_i into k prospective communities (C_{i1}, C_{i2}, ..., C_{ik}). Further, suppose that d_C is the value of a metric of interest, such as distance, averaged across all users in community C, and that smaller values for the metric are more desirable than larger ones. Finally, let $|C|$ be the number of users falling into community C. We can now define the *absolute performance gain* of splitting a community to be the improvement in the selected metric, weighted by the proportion of users in each of the prospective communities. That is:

$$\begin{aligned} absolute\ performance\ gain &= \sum_{j=1}^{k} \left(\frac{|C_{ij}|}{|C_i|} \times \left(d_{C_i} - d_{C_{ij}} \right) \right) \\ &= d_{C_i} - \sum_{j=1}^{k} \left(\frac{|C_{ij}|}{|C_i|} \times d_{C_{ij}} \right) \end{aligned}$$

We also define the *relative performance gain* of splitting a community to be the weighted average of the percentage improvement in the selected metric for each of the split communities, where the weights are the proportion of users falling into each of the communities. That is:

[2] We could consider all possible ways of combining context attributes together into communities and determine the performance experienced by users in each scenario. However, this presents a significant overhead when a contextual characteristic has a large number of context attributes, and thus deemed infeasible in an actual implementation.

$$
\begin{aligned}
relative\ performance\ gain &= \frac{absolute\ performance\ gain}{d_{C_i}} \\
&= \frac{\sum_{j=1}^{k} \left(\frac{|C_{ij}|}{|C_i|} \times \left(d_{C_i} - d_{C_{ij}} \right) \right)}{d_{C_i}} \\
&= \sum_{j=1}^{k} \left(\frac{|C_{ij}|}{|C_i|} \times \frac{d_{C_i} - d_{C_{ij}}}{d_{C_i}} \right)
\end{aligned}
$$

It may not be desirable to split communities if the relative performance gain that can be achieved is very small. We allow the operator to specify the minimum threshold of performance that must be exceeded for a community to be split. This threshold is called P_{MIN}. The choice of its actual value depends on how well the adaptation proxy is provisioned relative to its offered load. A well-provisioned system can have a low value for this parameter, which results in higher accuracy of the system's adaptation decisions. Ultimately, if the relative performance gain of making predictions from a restricted history exceeds that of using the full history by at least P_{MIN}, we decide that it is worthwhile to split the community. Otherwise, the status quo is maintained.

Figure 2 provides a simple example of a profiling experiment in a system that adapts image fidelity. In the example, it is assumed that users can select one of 10 possible fidelity levels available, numbered 1 to 10, for a specific image. Also, it us assumed that there is a single candidate contextual characteristic that has two possible context attributes: A and B. Thus, every user has context attribute A or B. The table on the left shows the desired fidelity level of an image for 6 users, and represents the status quo wherein users with both context attributes are grouped into a single community. Under the Mean prediction policy, the image would be served to members of this community at fidelity 7. The table also shows the distance of the prediction from the user's desired adaptation, and the context of each user. For the users of this community, we get an average distance of 2.17. The two tables on the right show the same set of users but the table on top groups together users with context attribute A (prospective community A) while the bottom table groups together users with context attribute B (prospective community B). The Mean policy predicts 4 for community A users and 9 for community B users. The average distance in each of the two prospective communities is 0.33. As the proportion of users falling into each of the communities is equal, the absolute performance gain from splitting is a 1.84 reduction in average distance. Also, we see that the relative performance gain of splitting is a 85% reduction in average distance.

Figure 3 illustrates the splitting process for a community. At the outset, an object is associated with a single community (node 1). The system creates more communities only if it believes that users accessing the object will get a worthwhile improvement in performance. From the default single community (node 1), users were first split based on bandwidth into 2 communities (node 2 for 56Kbps users and node 3 for 100Mbps users). The 56Kbps users were further split into 3 communities (node 4 for SmartPhone users, node 5 for PDA users and node 6 for GBook users). Ultimately, there are 4 communities that users accessing the object can be grouped into: 100Mbps users, 56Kbps-SmartPhone users, 56Kbps-PDA users and 56Kbps-GBook users.

3.2.3 Storage Considerations

Since splitting communities consumes additional storage, even if splitting a community is worthwhile because the performance gain exceeds the minimum threshold of P_{MIN}, this may not be feasible as all of the proxy's storage resources may have been consumed by previously split communities. We first provide an account of storage usage when splitting communities. Next, we describe the process used to optimize storage utilization on the adaptation proxy.

For the purposes of prediction, every community is associated with a single histogram that captures the desired adaptations of all members of the community. We refer to this as the *prediction histogram* for the community. In addition, a particular community that groups together users with x distinct contextual characteristics and y different context attributes, also maintains a set of y histograms - one for each context attribute - for purposes of context-based grouping. We refer to these as *context-grouping histograms*, and they capture the restricted history of the adaptations desired by members of the community with a specific context attribute. Thus, at the outset, we have $1 + y$ histograms associated with this community. Now, suppose that we split this community based on some contextual characteristic k, which is an enumeration of z context attributes. Each of the z new communities will be grouping together users with $x - 1$ distinct contextual characteristics and $y - z$ different context attributes. Thus, subsequent to the split, we have $1 + y - z$ histograms associated with each split community; in total, the split caused the creation of $z(1 + y - z) - (1 + y)$ additional histograms.

In order to optimize storage utilization on the proxy, we use a metric called *score* to indicate the relative importance of community splits. The score of a split takes into account how often users are being classified into each of the split communities as well as the absolute performance gain of having performed the split. The score metric provides meaningful comparison across communities (of the same or different objects) because it captures not only the benefit of having fine grain communities but also how that benefit is being utilized by users. We define the score of a split to be the product of the absolute performance gain of the split and the access rate of the associated object for users in the original community. That is:

$$
score = access\ rate \times absolute\ performance\ gain
$$

As different splits consume varying amounts of storage, we consider the score-per-byte when making comparisons. If there are previously split communities with a lower score-per-byte than that of the current split prospect, these are merged to make room. This process is repeated until enough space is available for the prospect community to split. If we do not find any previously split communities with a lower score-per-byte and space is still insufficient, the prospect split is abandoned.

If the split of a community can go ahead, we replace the old community with multiple new ones - one for every context attribute of the contextual characteristic that is the basis of the split. The prediction histogram of each of the new communities is seeded with the contents of the context-grouping histogram for the matching context attribute. This seeding process prevents the new communities from having to go through individual convergence phases and ensures that high quality predictions can be made from the outset. Alternatively, if a previously split community must be merged back, the prediction histogram of the split communities are aggregated together.

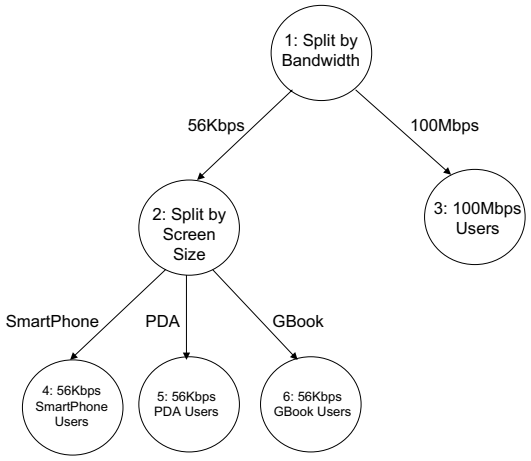

Figure 3: Illustration of how we split communities

Figure 4: Screenshots of page layout prototype on simulated displays of PDA (shown left) and SmartPhone (shown right). The toolbar at the top of the web page allows the user to manipulate the dimensions and layout of images.

3.3 Deployment Considerations

The CA-URICA architecture and grouping mechanism, as described in this and the previous section, is general in nature and can be applied to a wide range of different content adaptations, such as object fidelity and page layout transformations. An instantiation of this architecture for a specific class of adaptations (e.g., image fidelity adaptation) requires the system implementer or its operator to provide: the prediction policies to consider, a list of candidate contextual characteristics (along with context attributes) to explore, a choice of a metric to rate the quality of predictions, a value for T which controls when the profiling experiment is run, a value for P_{MIN} which places a minimum threshold on the performance required for communities to be split, and a set of transcoders to transform the content. While significant work is required to define the list of contextual characteristics and the other configuration parameters, it is important to note that once deployed, the system can adapt a wide set of content, without requiring the operator to create new rules to support a new data object. Instead, CA-URICA will leverage user-provided feedback to learn how to best adapt the new object. Moreover, we expect that a single adaptation proxy will be able to provide adaptation services for a large user community.

In many cases, there is a natural correlation between contextual characteristics. For example, we expect a relationship between a user's screen size and device type context. Despite the possible correlation, we do not require contextual characteristics to be independent of each other. When there is correlation between contextual characteristics that have a significant impact on adaptation performance, the FCS technique simply picks the alternative that results in the highest relative performance gain, i.e., our algorithm is greedy. Once the community has been split based on one context, the correlation implies that we have, to some extent, implicitly partitioned across the other contexts as well.

In our implementation, we conduct the profiling experiment after we have encountered at least T users for every context attribute of some contextual characteristic, where T is an operator specified parameter. However, other variations are possible. For instance, the test can be performed after some specified time has elapsed. Regardless of when the actual test is performed, it is important to have encountered a sufficient number of users for at least two context attributes. Our previous work and current experience have shown that prediction quality converges within the first 10 users. Thus, T should be at least 10 to ensure that the results of the profiling experiment are accurate.

As all client requests for objects are routed through the adaptation proxy, recently requested objects can be cached. It is important to note that the lifespan of cached objects is on the scale of minutes and hours while the lifespan of prediction information is on the scale of days, weeks, or even longer. However, as a web page changes over time, its usage semantics may alter. This can cause a change in adaptation requirements as well. When such changes occur, the previously accumulated history may no longer be relevant to making adaptation predictions. This can be addressed by periodically discarding some of the history for an object.

CA-URICA can determine community grouping and policy selection at the granularity of individual objects, such as specific images embedded in an HTML document. For efficiency, however, CA-URICA can be configured to determine community grouping and policy selection for sets of objects, such as all images in an HTML page, or all images on a Web site.

4. IMPLEMENTATION

We developed two context-aware URICA prototype adaptation systems to evaluate the effects of context on adaptation requirements and quantify the advantages of context-based grouping in CA-URICA. The first prototype provides page layout adaptation, which improves the user experience of browsing the web with mobile devices that have limited screen real estate. The second prototype provides fidelity adaptation, which allows users to save bandwidth as they browse the web. The rest of this section provides the implementation details of our two prototypes. Sections 5 and 6 describe the experiments we conducted with these prototypes and present the results of our evaluation, respectively.

4.1 Page Layout Adaptation Prototype

Our page layout prototype allows customization of the visual appearance of web pages on a user's device. Specifically, our prototype allows adaptation of the display size of images as well as their layout on the web page.

The prototype we describe here is a proof-of-concept and does

not provide all possible customizations. Also, the prototype currently functions on three simulated displays: a PocketPC SmartPhone, a PocketPC PDA and a Toyota GBook (an in-car browser).

When a web page is requested, the adaptation proxy also provides a Javascript component that renders a small toolbar at the top of the web page. If a user is not satisfied with the appearance of the web page on their device, they can use the toolbar to manipulate its appearance. Figure 4 provides screenshots of our page layout prototype on the PDA and SmartPhone simulated displays, shown left and right, respectively.

A user can resize individual images by clicking on them. The toolbar has a toggle for two modes: increase and decrease image size. If a user clicks on an image in increase mode, its height will increase by 30 pixels. In decrease mode, the height of the image decreases by 30 pixels for every click. The smallest height that an image can have is 30 pixels and the largest is 300 pixels. Width is always scaled to maintain image proportion. When no history is available, our prototype is configured to display images at their smallest height.

In our prototype, images can have 10 distinct heights. Thus the image size adaptation history for an image consists of the number of users who selected each of the 10 possible heights. Also, our choice of 30 pixels as the granularity of the resizing process, the minimum and maximum heights, as well as the number of distinct dimensions allowed are arbitrary design decisions for our prototype. In a real-world deployment, these can be configured by the system operator or specified by the content creator.

The toolbar can also be used to control the layout of images on the web page. We permit four distinct layouts: (1) all images on the same row; (2) two images on row 1 and one on row 2; (3) one image on row 1 and two on row 2; and (4) all images in different rows. When no history is available, our prototype is configured to display images with layout 1.

The image layout adaptation history for each page consists of the number of users who selected each of the 4 possible image layouts. The choice of these particular layouts is an arbitrary design decision of our prototype. A more sophisticated implementation may allow users to place any number of images in a given row. With a larger choice of layouts, and for web pages with more images, the pictorial representation of the layouts that we currently use is not sufficient. One way to address this is by having a drop-down list in the toolbar whose entries provide brief descriptions of the layouts. This is an implementation issue of the user interface that we plan to address in the future.

4.2 Fidelity Adaptation Prototype

Our second prototype provides fidelity adaptation of JPEG images, which provides a reduction in the amount of bandwidth consumed when browsing web pages. When a user requests a page, she is presented with fidelity adapted versions of all images on the page. If the user is unsatisfied with the fidelity provided for any image, she can request a refinement. Our fidelity adaptation prototype currently works with the Internet Explorer and Firefox web browsers on laptop devices, and also Pocket Internet Explorer on PocketPC PDAs.

The prototype is implemented such that when a JPEG image is first requested by a user, it is downloaded by the proxy and converted into a progressive JPEG image. This progressive image is then broken into 10 slices. When no history is available, the proxy serves the browser with the first slice of the image, which is approximately 10% of the file size of the original image. Progressive JPEG images have the property that even incomplete portions can be rendered on screen, albeit at a lower quality. If the user does not find the fidelity of the image served to be satisfactory, she can interact with the image (the image can be tapped on the PDA and clicked on the laptop) to request a refinement. For a refinement request, the proxy serves the subsequent slice of the image. On the client, this is concatenated with the previously downloaded data for the image and a higher fidelity version of the image is displayed to the user.

In our prototype, images can have 10 distinct fidelities and thus, the image fidelity adaptation history for an image consists of the number of users who selected each of the 10 possible fidelities. The choice of 10 fidelity classes is an arbitrary design decision that can be easily reconfigured to change the granularity at which the fidelity adaptation occurs.

4.3 Policies and Evaluation Metrics

For image layout adaptation, we use a prediction policy called MODE, which selects the most frequently requested layout for a page. The metric of evaluation is the percentage of times the policy incorrectly predicted the user's desired layout.

For image display size and fidelity adaptation, we implemented seven prediction policies: MEAN, MODE, MEDIAN, UPPER60, UPPER70, UPPER80 and UPPER90. The first 3 policies select the mean, mode and median of the entries in the history log, respectively. The UPPER family of policies construct a cumulative probability distribution function of user requests based on history and select the adaptation at which a specified percentage of user requests are satisfied. For example, out of 10 users, if 2 of them select image display size 2, 6 of them select display size 3 and 2 of them select display size 4, the UPPER60 policy will select image display size 3. The metric of evaluation is how far the prediction deviates from the adaptation required by the user. If the user selected a higher fidelity or larger display size than what a policy predicted, this indicates that the policy's prediction "undershot". When a policy undershoots, the difference between the user's desired adaptation level and a policy's prediction is captured in a metric called "Undershoot Distance". Alternatively, if the user selected a lower fidelity or smaller size than what a policy predicted, this indicates that the policy's prediction "overshot". When a policy overshoots, the difference between the policy's prediction and the user's desired adaptation level is captured in a metric called "Overshoot Distance". To quantify performance and make meaningful comparisons across policies, we use the distance metric defined earlier, which is simply the sum of "Undershoot Distance" and "Overshoot Distance." This cumulative metric penalizes policies equally for undershooting or overshooting. In a real-world deployment, user preferences may be such that either one could be more preferable. As such, a weighted sum may be more appropriate, where the weights reflect user preferences.

4.4 Discussion

One of the strengths of CA-URICA is that the user feedback that is utilized in making adaptation predictions is implicit. That is, the feedback is obtained during the normal course of operation of the adaptation system, and does not require solicitation from an altruistic user. However, image fidelity adaptation suffers from the problem that if we serve to a user an image that is of a higher quality than required, the user has little incentive to provide feedback. We refer to this as fidelity inflation, and address the problem by periodically performing a technique called *probing*, where we provide users with a lower fidelity version than what was predicted. As probing operates by undershooting deliberately, it is mildly obtrusive in that the user may have to provide some additional feedback when the prediction was accurate to begin with. Note that a prob-

lem similar to fidelity inflation does not occur in page layout or image display size adaptation since users have proper incentives to correct a bad layout or shrink a large image.

If the same image is shared between multiple pages, it may have different adaptation requirements on the individual pages. Also, the impact of contextual characteristics on adaptation requirements for that image may differ on various pages. This issue can be addressed by considering the page from which the user request for the object originates as a contextual characteristic. FCS can then be used to determine whether or not it is worthwhile to have distinct predictions for the image on different pages.

5. USER STUDIES

We conducted two user studies to investigate the effects of various contextual characteristics on content adaptation requirements, and to assess the benefits of grouping users into communities based on context. In the first study, we use our page layout prototype to explore the effects of context on image display size and layout adaptation, while the second study uses our fidelity adaptation prototype to explore the effects of context on image fidelity adaptation. Overall, we varied 4 contextual characteristics during the course of our studies: device type, screen size, network bandwidth and user location. Device type, screen size, and network bandwidth are examples of general contextual characteristics that are likely to affect a large set of web content. User location, on the other hand, is an example of a contextual characteristic that is meaningful only within the scope of specific web applications or domains, such as a map. A summary of the setup of our user studies is shown in Figure 5.

In each of the studies, users performed a fixed set of tasks as we varied the underlying context. During the study, we disabled the prediction mechanism of the prototypes. Thus, all users were served with the same default adapted versions, i.e., the prototypes did not take advantage of past interactions. In order to complete their tasks, users must interact with the prototype to obtain an appropriate adaptation. We collected traces of our user study subjects as they interacted with the two CA-URICA prototypes. In Section 6, we use these traces to evaluate the performance of our prototypes.

In the rest of this section, we first discuss our user study methodology. We then describe the user studies in detail.

5.1 Methodology

The participants in our user studies consisted of students at the University of Toronto. All of our studies were conducted in a controlled environment in a laboratory setting. On-line and written instructions were presented to the users. To mitigate any learning effects, all users were provided with a training period to familiarize themselves with the system prior to any data collection. Also, we did not allow users to perform the same component of a study over multiple contexts.

In each of our studies, we instructed users to complete a specific set of tasks. This was done for two reasons. First, in the real world, users browse the sites that they find interesting and have specific motivations for looking at content (e.g., read the latest news, or buy a product with a given feature). In our previous work, we have shown that user motivation has significant effect on the way in which an object should be adapted [26, 27]. However, this motivation cannot form spontaneously in a lab environment because test subjects lack an intrinsic interest in the content they browse. We overcome this problem by giving users tasks to perform that mirror the types of tasks they would perform in the real-world. Second, since all users are performing the same tasks, we know that any differences in the adaptations requested by users are the results of differences in individual user preferences and the context under which the study is being conducted. By giving users a common set of tasks involving the viewing of images, we can vary context between them in a controlled manner and quantify the effect that context has on adaptation requirements in different situations.

5.2 User Study 1: Investigating the effect of context on page layout adaptation

The objective of this user study is to evaluate the following two hypotheses:

- H_1: Screen size affects image display size adaptation requirements.
- H_2: Screen size affects image layout adaptation requirements.

We presented participants in the study with four web pages, each of which had three distinct images of postage stamps. At any given time participants were presented with only one page, and once they had moved to the next page they were not allowed to go back. We asked participants to play the role of a web site designer, and reconfigure each page to facilitate a comparison of two of its images and viewing a particular detail on the third image. Of the three images, the two being compared and the one being viewed in isolation changed between different pages. Participants could reconfigure the page by changing image display size and layout. We expected that participants would design the page such that they would be satisfied themselves if they were the one using it on a regular basis. Interviews performed subsequent to the study verified that this was indeed the motivation behind how participants designed the pages.

We obtained traces from 30 users. The users were randomly divided into three groups using different simulated displays: a PocketPC SmartPhone, a PocketPC PDA and a Toyota GBook vehicular terminal. Figure 4 shows a sample page on the simulated display of a PDA and SmartPhone. Vertical and horizontal scrollbars were placed by the web browser automatically when the page did not fit the screen. The traces obtained from the groups are referred to as SmartPhone, PDA and GBook, respectively.

5.3 User Study 2: Investigating the effect of context on fidelity adaptation

The objective of this user study is to evaluate the following three hypotheses:

- H_3: Network bandwidth affects image fidelity adaptation requirements.
- H_4: Device type affects image fidelity adaptation requirements.
- H_5: User location affects image fidelity adaptation requirements.

To evaluate H_3 and H_4, we developed a web site consisting of six web pages; each page containing an image of a car. At any given time participants were presented with only one page, and once they had moved to the next page they were not allowed to go back. We configured the prototype so that it loaded images at the lowest fidelity. Participants could improve image fidelity by tapping or clicking on the image. We asked users to record the license plate numbers for cars that have license plates. Only two cars in the site have license plates, but this information is not provided to the users. We expected that users will require different fidelity levels for different images because it is possible to determine that

	Contextual Characteristics	Context Attributes	Number of Users
User Study 1: Page Layout Prototype			
Postage Stamps	Screen Size	SmartPhone Display	10
		PDA Display	10
		GBook Display	10
User Study 2: Fidelity Adaptation Prototype			
Cars	Network Bandwidth, Device Type	Laptop-56Kbps	25
		Laptop-100Mbps	25
		PDA-56Kbps	25
Map	Location	Location 1	40
		Location 2	40
		Location 3	40

Figure 5: Summary of setup of User Studies

a car does not have a license plate with a low-fidelity image, but higher levels of fidelity are required for cars with license plates in order to identify the license plate number.

We asked three different groups of 25 students each to conduct the above tasks. The first and second groups performed their tasks on a laptop computer connected over a 56Kbps and a 100Mbps network link, respectively. The third group completed their tasks using a PDA connected over a 56Kbps link. We refer to the traces obtained from these groups as laptop-56Kbps, laptop-100Mbps and pda-56Kbps, respectively.

To evaluate H_5, we designed a map web site. We presented participants with a map of the campus of the University of Toronto. The map is actually composed of a 6 X 6 grid of images. Upon loading the web site, all images in the grid are visible at the lowest fidelity. It is possible to refine individual images in the 6 X 6 grid by clicking on them. Users are asked to record the names of all the buildings in the path between specified start and end points, which are well-known locations at the University (e.g., library to main subway station). To complete the task, users need to increase the fidelity of images that are on the path between the source and the destination in order to record the building names. Users behave in a uniform way because they have enough familiarity with the map to determine which images are relevant to the task even at low fidelity, but require higher fidelity to recognize the names of buildings.

We had three groups of 40 users each. Each group was given a distinct start location, where the paths of users in different groups did not intersect. We refer to the traces obtained from these groups as L1, L2 and L3, respectively.

6. EVALUATION

In this section, we provide an evaluation of CA-URICA augmented with FCS. We use the traces that we collected during our two user studies to evaluate our page layout and fidelity adaptation prototypes. It is important to note that our purpose is not to conclusively show, once and for all, what context matters. In fact, we expect that the context that has the most impact on adaptation requirements will vary across content and different types of adaptation. Our goal is to evaluate the ability of our profiling approach to identify the context that is important in any given situation.

Our exposition mirrors the model shown in Figure 2. At the outset, all users are classified into the original community regardless of their context. During the profiling experiment, the system considers whether a performance improvement can be achieved for a group of users sharing a context attribute if they are separated from the original community. It should be noted that the overall improvement that will be observed depends on the composition of users in the original community. For instance, even when a particular contextual characteristic C has significant bearing on adaptation requirements, if (prior to any splits) an overwhelming majority of users accessing an object have the same context attribute in C, we would not see a significant improvement for splitting the community by C. The composition of communities that we used in our experiments is based on the data collected during our user studies, and represents an intermediate case where there is a similar number of users with each context attribute.

6.1 Impact of Context on Page Layout

We start by considering the influence of screen size on image display size adaptation. Figure 6 provides the results from a profiling experiment where the system is considering whether to partition an existing community based on screen size. The original community consists of 30 users in the SmartPhone, PDA and GBook datasets, while each of the prospective community consists of the 10 users with the same screen size. We show the average distance from desired size across all images, averaged across all users in a community, when predictions are made using the Mean policy. The first two bars show the overall performance of all users, while the remainder show the performance of users with the specified screen size. We see that grouping users by screen size results in a 29% reduction in distance overall. The significant reduction in distance implies that splitting users based on the screen size contextual characteristic will provide significant improvements in performance for image display size adaptation. Thus, we see that for the Postage Stamps web site: (R1 [3]) screen size affects image display size adaptation.

Next, we consider the influence of screen size on image layout adaptation. Figure 7 provides the results from a profiling experiment where the system is considering whether to partition the existing community based on the screen size context of users. As before, the original community consists of 30 users in the SmartPhone, PDA and GBook datasets, while each of the prospective community consists of the 10 users with the same screen size. We show the percentage of incorrect layout predictions, averaged across all users in a community, when predictions are made using the Mode policy. We see that grouping users by screen size reduces the percentage of layout mispredictions from 43% to 34% overall, an improvement in the performance of the adaptation system of 21%. The significant reduction in the percentage of mispredictions implies

[3]The R (Result) in section 6 correspond to H (Hypothesis) in section 5

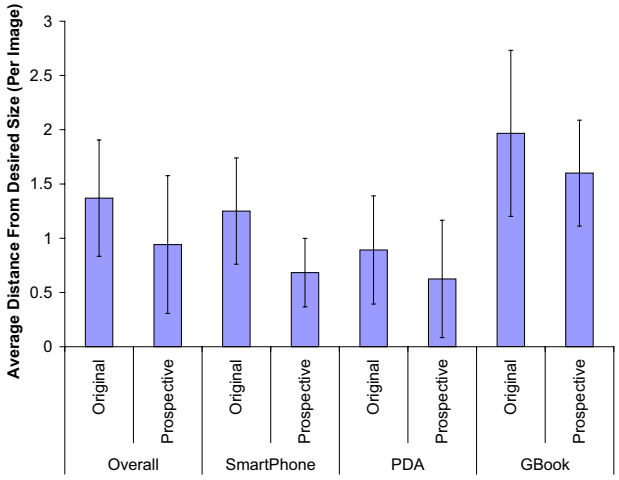

Figure 6: Grouping users by screen size improves prediction quality for image display size adaptation in the Postage Stamps web site. We present the average distance from desired size across all images, averaged across all users in a community, when predictions are made using the Mean policy. Original shows performance when we group SmartPhone, PDA and GBook users together into a single community. Prospective illustrates performance when users are grouped by screen size. The first two bars show the overall performance of all users, while the remainder show the performance of users with different screen sizes.

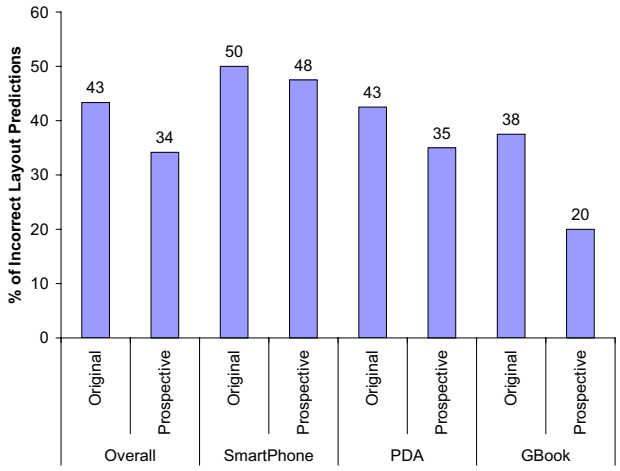

Figure 7: Grouping users by screen size improves prediction quality for page layout adaptation in the Postage Stamps web site. We present the percentage of incorrect layout predictions, averaged across all users in a community, when predictions are made using the Mode policy. Original shows performance when we group SmartPhone, PDA and GBook users together into a single community. Prospective illustrates performance when users are grouped by screen size. The first two bars show the overall performance of all users, while the remainder show the performance of users with different screen sizes.

that splitting users based on the screen size contextual characteristic will provide significant improvements in performance for page layout adaptation. Thus, we see that for the Postage Stamps web site: (R2) screen size affects image layout adaptation.

6.2 Impact of Context on Fidelity

We start by considering the influence of network bandwidth on image fidelity adaptation. Figure 8 provides the results from a profiling experiment where the system is considering whether to partition an existing community based on the network bandwidth context of users. The original community consists of the 75 users in the laptop-56Kbps, laptop-100Mbps and pda-56Kbps datasets. The 100Mbps prospective community consists of the 25 users in the laptop-100Mbps dataset, while the 56Kbps prospective community consists of 50 users: 25 users from the laptop-56Kbps and pda-56Kbps datasets each. We show the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. We see that grouping users by network bandwidth results in a 14.5% reduction in distance overall. The significant reduction in distance implies that splitting users based on the bandwidth contextual characteristic will provide significant improvements in performance for image fidelity adaptation. Thus, we see that for the Cars web site: (R3) network bandwidth affects image fidelity adaptation.

Figure 9 shows the average number of Kilobytes consumed by users in the previous experiment. As previously noted, predictions are made using the Mean policy. While the Cars web site consists of 1473KB of image data in total, the average amount of bandwidth required by users in the Overall, 100Mbps and 56Kbps datasets is 402KB, 535KB and 336KB, respectively. We see that grouping users by bandwidth results in a 8% reduction in the amount of bandwidth consumed overall. However, the reduction in bandwidth consumption does not affect all users in the same way. Users with 100Mbps connectivity end up consuming slightly more bandwidth (an increase of 4.6%) but they also have to interact with the system less frequently (a reduction of 26%). Alternatively, users with 56Kbps connectivity consume far less bandwidth (a reduction of 16%). For these users, the performance of the system approaches the optimal amount of bandwidth consumption.

Next, we consider the influence of device type on image fidelity adaptation. Figure 10 provides results from a profiling experiment where the system is considering whether to partition an existing community of users with 56Kbps network connectivity based on the device type context of users. The original community consists of 50 users in the pda-56Kbps and laptop-56Kbps datasets, while each of the prospective community consists of the 25 users with the same device type. We show the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. We see that grouping users by device type results in a 6% reduction in distance overall. The lack of a significant reduction in distance implies that splitting users based on the device contextual characteristic will not provide significant improvements in performance for image fidelity adaptation. Thus, we see that for the Cars web site: (R4) device-type does not affect image fidelity adaptation.

Finally, we consider the influence of location context on image fidelity adaptation in the map web site. Figure 11 provides the results from a profiling experiment where the system is considering whether to partition an existing community based on the location context of users. The original community consists of 120 users in the L1, L2 and L3 datasets, while each of the prospective community consists of the 40 users with the same location. We show the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. We see that grouping users by location results in a

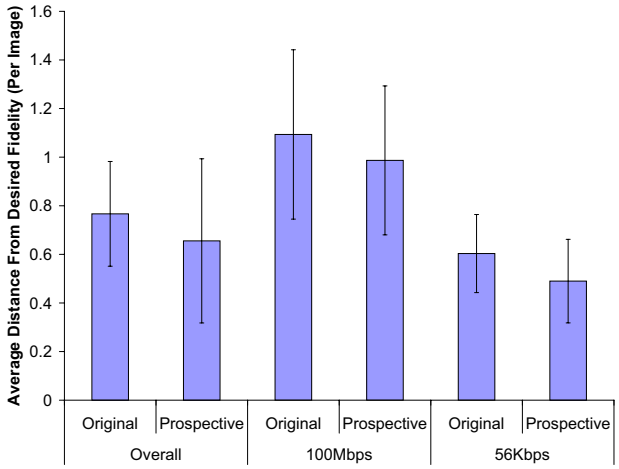

Figure 8: Grouping users by bandwidth improves prediction quality for image fidelity adaptation in the Cars web site. We present the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. Original shows performance when we group 100Mbps (laptop-100Mbps), and 56Kbps (laptop-56Kbps and pda-56Kbps) users together into a single community. Prospective illustrates performance when users are grouped by bandwidth. The first two bars show the overall performance of all users, while the remainder show the performance of users with different network bandwidth.

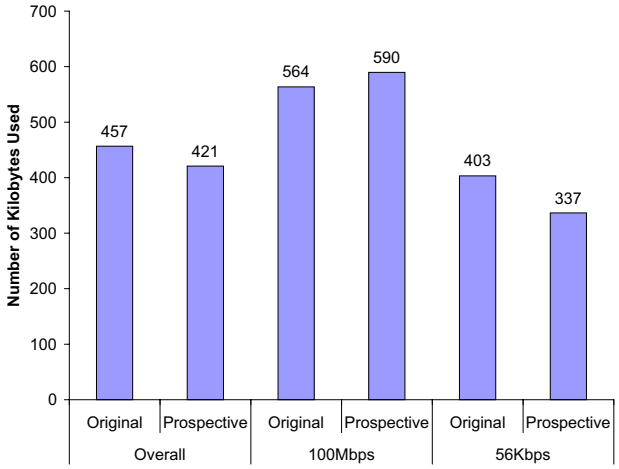

Figure 9: Effect of grouping users by network connectivity on the amount of bandwidth consumed for image fidelity adaptation in the Cars web site. We show the average number of Kilobytes consumed, when predictions are made using the Mean policy. Original shows performance when we group 100Mbps (laptop-100Mbps), and 56Kbps (laptop-56Kbps and pda-56Kbps) users together into a single community. Prospective illustrates performance when users are grouped by bandwidth. The first two bars show the overall performance of all users, while the remainder show the performance of users with different network bandwidth.

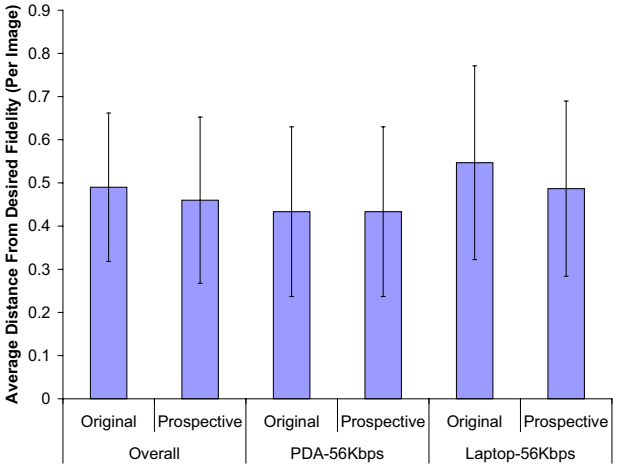

Figure 10: Grouping 56Kbps users by device does not significantly improve prediction quality for image fidelity adaptation in the Cars Web site. We present the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. Original shows performance when we group pda-56Kbps and laptop-56Kbps users together into a single community. Prospective illustrates performance when users are grouped by device type. The first two bars show the overall performance of all users, while the remainder show the performance of users with different device types.

79% reduction in distance overall. The significant reduction in distance implies that splitting users in the map web site based on the location contextual characteristic will provide significant improvements in performance for image fidelity adaptation. Thus, we see that for the Map web site: (R5) User location affects image fidelity adaptation.

6.3 Summary of Results

During the course of our evaluation, we observed that users performing the same tasks under different context had different adaptation requirements, and also that the adaptation requirements for users with the same context were similar. Our experiments showed that grouping users into communities based on context can lead to significant improvements in overall performance. However, we also observed that the influence of context on adaptation requirements can vary based on the content being adapted, as well as the type of adaptation being performed. Thus, we find that CA-URICA successfully provides fine grain adaptation for content.

7. RELATED WORK

Adaptation is a well-studied topic in mobile and pervasive computing [1, 13, 14, 17, 19, 28, 33, 34, 39]. Numerous systems have been built by researchers in academia and industry to explore various aspects of the problem.

There are two general high-level techniques for automatic adaptation policy generation: rule-based and constraint-based adaptation. Several systems using these techniques are described below. In contrast to these techniques, CA-URICA dynamically determines how to adapt based on the adaptation requirements of users in the past. This obviates the need to have a human designer specifying rules or constraints for every content object and context,

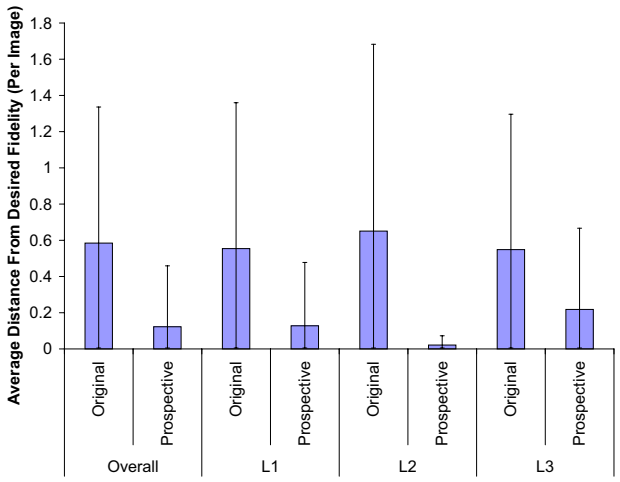

Figure 11: Grouping users by location improves prediction quality for image fidelity adaptation in the Map web site. We present the average distance from desired fidelity across all images, averaged across all users in a community, when predictions are made using the Mean policy. Original shows performance when we group L1, L2 and L3 users together into a single community. Prospective illustrates performance when users are grouped by location. The first two bars show the overall performance of all users, while the remainder show the performance of users with different locations.

and also provides better performance than having a small number of general-purpose rules. Also, users do not have to provide explicit preferences on a per-object basis.

Odyssey [29], an adaptation system built on top of the Coda file system [21], allowed applications and the Operating System to collaborate in order to customize applications and content in the face of dynamically changing resources on mobile devices. Odyssey identified that there are some aspects of adaptation that are best centralized (e.g. monitoring resource levels, arbitration in the face of concurrency) while others should be left to the individual applications (e.g. selecting fidelity under different resource availability conditions).

Puppeteer [12], a system for adapting applications and content for the mobile environment, took advantage of the exported APIs of component-based applications to customize the behavior of applications based on their environment without modifying the source code of the original application. A similar approach can be used to allow CA-URICA's feedback mechanism to be incorporated into a plug-in component, which is developed using an application's API.

Fox et al. proposed a proxy-based approach [15, 16] to content adaptation wherein a proxy is interposed in the network path between the client and the server. The system allowed users to specify preferences such as desired download time, image resolution and color depth. CA-URICA has a similar proxy-based architecture. However, CA-URICA attempts to predict the adaptation desired by users based on the content's usage semantics and the user's context. Moreover, users are allowed to change the adaptation provided by the system, and this feedback is used to refine the adaptation decision.

IBM's Web Transcoding Publisher [3] provides a framework for taking original content and transcoding it to better suit the requirements of the client device requesting the content. Microsoft's ASP.NET mobile controls [25] also provide similar functionality. In both cases, the decision of how to adapt content for individual devices is based on static device profiles that contain rules of how different types of content should be adapted for particular devices.

Smith et al. built a system [35] that analyzed images and classified them into one of several categories. Rules, taking into account the image type, were used to make the adaptation decision. This was extended into the Infopyramid system [36] in which scores were assigned to various transcoded versions of content based on how much a particular transcoding scheme would degrade the original object. Constraints such as device storage capacity and load time could be specified, and the system attempts to maximize the value of the content without violating any constraints.

There have been a number of research efforts that attempt to restructure the layout of web pages [9, 10, 31, 37]. Wireless Application Protocols (WAP) such as Wireless Markup Language (WML) and Handheld Device Markup Language (HDML) are popular approaches to customize the appearance of web content for mobile devices [38]. These protocols can be used to split an HTML page into small cards such that each card can fit into a single screen. Links are made between cards to enable browsing of the page. The two main weaknesses [4] of this approach are that we lose the layout and presentation style of the original web page, and that navigating the cards can be a cumbersome process requiring a large number of selections to retrieve desired information.

Digestor [2] is an automatic web page reauthoring system, in which an HTTP proxy manipulates the layout of pages in order to improve their presentation on mobile devices. The system uses a fixed set of rules that are applied to content. For example, a web page can be outlined, where only the contents of HTML section headers are displayed initially as hyperlinks. Links can be navigated to retrieve a page that contains the contents of the corresponding section.

Power Browser [5, 6, 7] is a browser for mobile devices that initially provides a summary view of the entire web page. The summary view is generated using heuristics that attempt to gauge the relative importance of words in blocks such as paragraphs. The user can interact with each summarized item in order to get an expanded view of the block.

Lum et al. developed a context aware system [23] that tailored the adaptation performed based on a user's context. Users provide the system with their preferences apriori, such as their sensitivity to color, and this is used to score various possible customizations. The customization that is ultimately presented to users is decided based on a negotiation algorithm that considers the user's preference scores as well as real-time context characteristics such as network connectivity.

Context Weaver [11] is a middleware system that facilitates the use of context information by applications. Applications do not have to request information from explicit sources of context data. Rather, they specify the type of context information that they require, and Context Weaver meets the application's requirements using the most appropriate source of information.

URICA has been previously presented in [26, 27]. This earlier work showed that it is possible to leverage user interaction (to correct adaptation decisions) to learn how to adapt content based on usage semantics. In contrast, this paper shows that it is possible to leverage user interaction to determine which contextual characteristics have the most impact on adaptation requirements, and should therefore be the basis of grouping users into communities. In addition, while the previous work only dealt with image fidelity adaptation, this paper also provides a description and evaluation of a prototype system for page layout adaptation.

There is a large body of work in machine learning [18, 24, 40] re-

garding clustering and classifying data points. If we use a standard clustering algorithm to group users together in CA-URICA, we are faced with the challenging question of how to classify users into communities. Using context information as the basis for grouping users makes classification natural - users are classified into the community that matches their context.

8. CONCLUSIONS

We showed that not all contextual characteristics equally affect adaptation requirements, and that by leveraging user interaction, it is possible to automatically determine the contextual characteristics that have the most impact on the adaptation requirements of an object. Moreover, we showed that grouping users into communities based on context improves the quality of the adaptation system's predictions by up to 79%.

We described Feedback-driven Context Selection (FCS), the automatic technique that context-aware Usage-awaRe Interactive Content Adaptation (URICA) uses to group users into communities. FCS takes advantage of user interaction to determine those contextual characteristics that have the most impact on the adaptation requirements of an object, and therefore should be the basis of grouping users into communities. Initially, an objects that is being adapted (e.g., an image in a Web page) is associated with a single community. FCS creates additional fine grain communities if it determines that splitting users into multiple communities based on a given contextual characteristic would result in a significant improvement in the quality of the system's adaptation decisions.

We implemented two context-aware URICA prototypes that provide page layout and image fidelity adaptation. The page layout prototype supports browsing on devices with limited screens by allowing users to change the layout and dimensions of images embedded in HTML pages. The fidelity adaptation prototype reduces bandwidth consumption for web browsing by providing lower fidelity JPEG images, and allowing users to refine the fidelity of individual images. Our prototypes provide automatic adaptations that reflect both the user's context and the content's usage semantics. This kind of customization was previously available only in manually adapted content.

Context-aware URICA achieves near-optimal adaptation performance for content that is being used for a single purpose. When content is being used for multiple purposes, we continue to see benefits over no adaptation; however, optimal performance is not attained. In the future, we plan to develop algorithms that will provide optimal adaptation performance when content is being used for multiple purposes.

REFERENCES

[1] R. Bagrodia, W. W. Chu, L. Kleinrock, and G. Popek. Vision, issues, and architecture for nomadic computing. *IEEE Personal Communications*, 2(6):14–27, Dec. 1995.

[2] T. W. Bickmore and B. N. Schilit. Digestor: Device-independent access to the World Wide Web. *Computer Networks and ISDN Systems*, 29(8–13):1075–1082, 1997.

[3] 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.

[4] G. Buchanan, S. Farrant, M. Jones, H. W. Thimbleby, G. Marsden, and M. J. Pazzani. Improving mobile internet usability. In *World Wide Web*, pages 673–680, 2001.

[5] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Accordion summarization for end-game browsing on pdas and cellular phones. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220, New York, NY, USA, 2001. ACM Press.

[6] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: text summarization for web browsing on handheld devices. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 652–662, New York, NY, USA, 2001. ACM Press.

[7] O. Buyukkokten, H. G. Molina, A. Paepcke, and T. Winograd. Power browser: Efficient web browsing for PDAs. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI'00*, 2000.

[8] 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.

[9] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, 2003. ACM Press.

[10] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, 2003. ACM Press.

[11] N. H. Cohen, J. Black, P. Castro, M. Ebling, B. Leiba, A. Misra, and W. Segmuller. Building context-aware applications with context weaver. Technical report, IBM Research, Oct. 2004.

[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] D. Duchamp. Issues in wireless mobile computing. In *Proceedings of Third Workshop on Workstation Operating Systems*, pages 1–7, Key Biscayne, Florida, Apr. 1992.

[14] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, pages 38–47, Apr. 1994.

[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, Aug. 1998.

[17] 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.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[19] R. H. Katz. Adaptation and mobility in wireless information systems. *IEEE Personal Communications*, 1(1):6–17, 1994.

[20] T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1), Jan. 2002.

[21] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, Feb. 1992.

[22] 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.

[23] W. Y. Lum and F. C. Lau. A context-aware decision engine for content adaptation. *IEEE Pervasive Computing*, 1(3):41–49, July 2002.

[24] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[25] Microsoft. Asp.net mobile designer. `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mwdesign/html/mworiMobileInternetToolkitDesigner.asp`.

[26] I. Mohomed, J. C. Cai, and E. de Lara. Urica: Usage-aware interactive content adaptation for mobile devices. In *Proceedings of EuroSys'06*, Leuven, Belgium, 2006.

[27] I. Mohomed, A. Chin, J. C. 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] 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.

[31] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis. Automatic detection of fragments in dynamically generated web pages. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 443–454, New York, NY, USA, 2004. ACM Press.

[32] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Fifteenth ACM Symposium on Principles of Distributed Computing*, Philadelphia, Pennsylvania, May 1996.

[33] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 2001.

[34] 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.

[35] 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.

[36] 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.

[37] K. Sumita, K. Ono, and S. Miike. Document structure extraction for interactive document retrieval systems. In *SIGDOC '93: Proceedings of the 11th annual international conference on Systems documentation*, pages 301–310, New York, NY, USA, 1993. ACM Press.

[38] WAP Forum. Wireless application protocol architecture specification, Apr. 1998. Available at: http://www.wapforum.org/what/technical/arch-30-apr-98.pdf.

[39] M. Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, July 1993.

[40] I. Witten and E. Frank. *Data Mining*. Morgan Kaufmann, 2000.