Community-Driven Adaptation: Automatic Content Adaptation in Pervasive Environments

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

Abstract

Mobile devices are increasingly being used to access Web content but lack the resources for proper presentation to the user. To address this problem, content is typically adapted to be more suitable for a mobile environment. Community-Driven Adaptation (CDA) is a novel approach to automatic content adaptation for mobile devices that adapts content based on feedback from users. CDA groups users into communities based on common characteristics, and assumes that users of the same community have similar adaptation requirements. CDA learns how to adapt content by observing how members of a community alter adapted content to make it more useful to them. Experiments that consider the idealized case, where all users perform the same task, show that CDA can reduce wastage of network bandwidth by up to 90% and requires less user interaction to correct bad adaptation decisions compared with existing approaches to automatic content adaptation.

1. Introduction

Mobile devices, such as cell phones and network-enabled personal digital assistants (PDAs) are fast becoming the platform of choice for accessing Web content, and are expected to play a leading role in the pervasive or ubiquitous-computing [23, 28] environments of the future. However, most Web content is intended for consumption on powerful desktop computers, and has to be adapted to meet the limited resources of mobile devices [1, 4, 8, 9, 15, 17, 22]. This problem is exacerbated by an astonishing variety of mobile devices with a wide array of capabilities (in terms of network bandwidth, display size, user interface, computing power, storage, battery life, security and reliability), and a similar multiplicity of data formats, communication protocols, and supported services.

Systems that adapt content on-the-fly to specific devices [4,6,10,18,20,25,26], have surfaced as a promising approach for coping with the resource constraints and high degree of heterogeneity in pervasive-computing environments. The main challenge in automatic content adaptation is designing adaptation policies that make optimal use of resources and maximize user satisfaction. This is a non-trivial problem as it is possible to adapt content across many different dimensions, such as fidelity, page layout and modality (e.g., transforming text into speech). Even after deciding how to adapt content, the question remains as to how much adaptation to perform. Too much adaptation is bad as it leaves users with unusable content or may require frequent user interaction to correct bad adaptation decisions – a major inconvenience for the user. Too little adaptation is also undesirable because it wastes valuable resources without producing additional benefit for the user.

This paper presents Community-Driven Adaptation (CDA), an automatic content adaptation technique that adapts content based on feedback provided by users. CDA defines a community as a set of users with similar requirements for adaptation. CDA groups users into communities based on common characteristics (e.g., device type or preferences) and makes a decision on how to adapt content based on past encounters with members of that community. This is done on the premise that adapted content that is suitable for some members of the community is likely to be acceptable to other members as well. At the outset, CDA makes an initial decision on the adaptation to be performed on the content and then monitors users as they modify the adaptation decisions to make the content more appropriate for their tasks (e.g., increase image fidelity to allow a detailed comparison between two products). The user modifications constitute an implicit feedback mechanism that CDA uses to improve its adaptation predictions for future user accesses. All feedback is obtained during the course of normal application usage, and users do not have to perform any additional interaction other than what is required for completing the task. To reduce the burden that being an active part of the adaptation process places on individual users, CDA leverages the aggregate experiences of large numbers of users accessing common content. Therefore, we do not expect that any given user will of-

ten find herself in the case where there is little historical data. For CDA to be effective, a critical mass of users accessing the same content is required. However, when this is not the case, users do not experience any worse performance than existing techniques, as CDA defaults into a rule or constraint-based approach when sufficient historical data is not available.

CDA is a significant improvement over existing rule-based [4, 14, 24–26] and constraint-based [7, 18, 19, 26] approaches for content adaptation. Rule-based systems rely on high-level rules to guide the adaptation process (e.g., convert images larger than 50 KB to progressive JPEG images). Constraint-based adaptation extends rule-based adaptation to encode tradeoffs between possible adaptation strategies. Rule-based and constraint-based approaches are limited to performing semantic-blind adaptation that apply to a large class of content (e.g., all thumbnail images are served at 60% resolution while full-sized images are served at 30% resolution), whereas CDA adapts content based on the data semantics and its relationship to the user's task (e.g., images central to a task are presented with higher detail than irrelevant objects).

CDA is a general technique that can be used to automatically adapt content in the face of resource constraints that are typical in ubiquitous-computing environments. On mobile devices that have limited screen real-estate, CDA can be used for page-layout adaptation. CDA can also be meaningfully applied to fidelity and modality adaptation of content such as images, audio or video. Configuring application parameters is another example of an adaptation domain where CDA can be used to improve user experience. In general, CDA can be useful in any situation where users share preferences in the kinds of adaptations that they require.

This paper establishes the efficacy of CDA. We deliberately sidestep the important problem of grouping users into communities, and instead quantify the effectiveness of CDA by considering an idealized environment where all users perform the same task on the same device, and therefore belong to one highly-cohesive community. Such singleness of purpose among the members of a community is unlikely to exist in practice. Therefore, the results presented in this paper are an upper bound on the perfromance of CDA.

We gathered traces of users browsing adapted image-rich web pages over a bandwidth-limited network link. The participants in the case study were assigned a common set of tasks to perform over a number of web sites. All the images on the web sites were initially served at their lowest fidelity, and we tracked users as they increased the fidelity of individual images until they were good enough for the task at hand. This allowed us to determine the lowest fidelity of each image deemed sufficient for the tasks by every user in the community. We then used trace-driven simulation to evaluate the performance of various rule-based and CDA adaptation policies. Experimental results show that CDA policies outperform rule-based ones, resulting in up to 90% reduction in network bandwidth wastage and up to 50% reduction in user requests to improve fidelity. The traces show that users select similar fidelity levels for the same task (which we define as *locality*), confirming our initial hypothesis that users in the same community have similar requirements for adaptation. Moreover, the traces show that the fidelity level for an object is dependent on its relevance to the user's task, and not on image size, purpose (thumbnail vs. regular image), or location (across web sites).

The rest of this paper is organized as follows. Section 2 introduces Community-Driven Adaptation and describes a general architecture for implementing this concept. Section 3 details a system that we developed for gathering traces of users accessing adapted web images, and Section 4 presents the experimental results from our trace-driven simulation. Section 5 describes related work on automatic content adaptation and recommendation-based systems. Finally, Section 6 concludes the paper and describes avenues for future work.

2. Community-Driven Adaptation

In this section, we first describe the concept of Community-Driven Adaptation (CDA) and provide an architecture that implements this concept. Next, we discuss the adaptation process and the role of feedback. Finally, we discuss practical deployment issues.

CDA is an automatic content adaptation technique that utilizes feedback provided by users. CDA recognizes that users (as opposed to the adaptation system) are best qualified to determine whether content that has been adapted is appropriate for a particular task. When serving an item for the first time, CDA makes an educated guess about the best way to adapt the content and behaves in a way that is similar to existing constraint-based content adaptation systems. CDA's strength comes into play when the user determines that the adapted content is not satisfactory and instructs the application to correct the situation (e.g., make content more readable by removing an irrelevant toolbar, convert text to speech, or increase or reduce a video's frame rate). The user modifications constitute a feedback mechanism that CDA uses to improve its adaptation predictions for future accesses.

CDA minimizes the need for user intervention by grouping users into communities such that members of a community share requirements for adaptation. CDA assumes that by observing the manner in which individual users interact with adapted content, it is possible to improve the effectiveness of content adaptation for the community as a whole.

We expect that in most cases, simply discernable characteristics such as device type, network bandwidth, location, battery power level, screen real-estate, user preferences, and available I/O modalities will be sufficient to effectively group users into communities. For example, we expect that users browsing high-resolution images on a PDA with a low-resolution display will require similarly adapted versions of

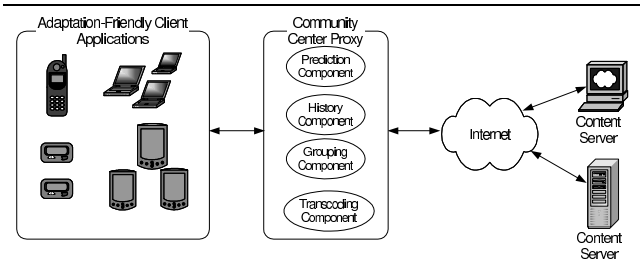

Figure 1. CDA architecture.

the images. However, in some situations, this will not suffice and the user's actual purpose vis-a-vis the content will have to be examined. This is a non-trivial problem and we foresee that complex characteristics such as user-supplied profiles, commonality in access history or past content adaptation requirements and context information such as user location can be used to discern the user's motivations with respect to the content. In this paper, we do not give further consideration to how communities can be formed and this is left for future work.

2.1. CDA Architecture

CDA uses a proxy-based architecture (shown in Figure 1) consisting of three components: *adaptation-friendly client applications*, *community center proxy* and *content servers*. Adaptation-friendly client applications allow users to interact with the community center proxy to refine adaptation decisions to tailor the adapted content to the requirements of the task. The community center proxy is at the heart of the CDA model. It mediates all accesses to content and serves as an aggregation point for user requests. Content servers are arbitrary data repositories that store the content that the user is accessing and are unaware of the adaptation process.

2.1.1. Adaptation-Friendly Client Applications CDA assumes that adaptation-friendly client applications provide users with the necessary facilities to make content more appropriate to their tasks. An adaptation-friendly client application should ideally 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 interface for correcting adaptation decisions, (3) it should propagate adaptation corrections to the community center proxy, and (4) for adaptation corrections that require the community center 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 and updating the application's state accordingly. 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.

The interface that the application provides for correcting adaptation decisions depends on the data type being adapted, and the adaptations supported by application and the community center proxy. For example, an application that supports the adaptation of streaming video could provide a control that affects the size of the window that plays the video, the image's resolution, frames per second, or position of the stream where playback starts. For document layout adaptation, the application may 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 require making 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 [6].

Given that CDA aggregates the results of all users that provide feedback, even clients that do not wish or are unable to run adaptation-friendly client applications can still benefit from the results, provided that a sufficient number of other users are using an adaptation-friendly client application.

2.1.2. Community Center Proxy The community center proxy mediates requests between adaptation-friendly client applications and content servers. It consists of four components: *prediction*, *history*, *grouping*, and *transcoding*. The prediction component determines the adaptations to perform on the content requested by the user. It uses a combination of rules and constraints augmented by historical data. The history component keeps track of user requests, the initial adaptations performed by the system, and any explicit modifications that users make to adaptation decisions. These modifications are an indication that the prediction component failed to adapt to the extent desired by the user, and are taken into consideration by the prediction component to improve its prediction strategy for future requests. The grouping component classifies users into communities based on a variety of factors such as user-supplied profiles, commonality in access history or content adaptation requirements, device type, application being used, and context information such as user location. Finally, the transcoding component coordinates a rich set of content transcoders that perform the actual content transformations prescribed by the prediction component and cache them for future user requests.

2.1.3. Content Servers Content servers are standard unmodified data repositories such as web sites, databases, and media servers. CDA does not require any additional functionality for this component.

2.2. Adaptation Process

The adaptation process for CDA operates as follows. A user first requests content through the adaptation-friendly client application. The adaptation-friendly client application

forwards the user's request to the community center proxy. The grouping component within the community center proxy determines the community that this user belongs to. At the outset, when the community center proxy has not gathered sufficient historical data for a specific data object within a community, the prediction component adapts based on rules and constraints. As such, when there is no history, the system performs no worse than the state-of-the-art. However, as the system accumulates historical data, we can expect the performance of the system to surpass that achieved by existing techniques. Once the adaptation decision is made, the community center proxy checks whether it has an appropriately adapted version of the desired content. If so, the content is served to the user. If not, the content is first fetched from the content server, and then appropriately adapted by the transcoding component. The adapted version is then placed in the cache, and a copy is returned to the client. If the user determines that the provided version is not appropriate for the task, she can then use the interface provided by the application to apply a corrective measure. The user's correction is then sent to the community center proxy where it becomes part of the history of the particular object being adapted for the user's community. If the corrective measure requires the transmission of an atomic object, such as an image from the community center proxy, 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 community center proxy negotiate to change the necessary properties of the stream.

2.3. User Feedback

The corrective measures employed by the user constitute a feedback mechanism that CDA uses to improve its adaptation predictions for future accesses. User feedback can be either explicit or implicit. Explicit feedback requires extra effort from the user for rating the adaptation decision while implicit feedback is obtained in the process of normal application operation. In both cases, the feedback can be deemed positive or negative, with respect to the user's reaction to the adaptation decision made by the system. If explicit feedback is available, there is no ambiguity in determining the user's preferences. Explicit positive or negative feedback fully assures the adaptation system of the correctness or lack thereof for an adaptation decision.

Implicit feedback does not entail any explicit rating of the adaptation decision. Implicit negative feedback occurs when a user is not satisfied with the adapted content that was served and requests a change in order to complete a task. In this case, it is clear that the previous adaptation was insufficient and there is no ambiguity in determining the user's preferences. Implicit positive feedback occurs when the user does not interact with the content and in this case, it is unclear whether the user was satisfied with the adaptation or simply decided to give up performing the task.

A major problem with systems that require explicit feedback is that users must be somehow enticed to provide additional information, which is not relevant to the performance of their task. Apart from the added burden that this imposes, the system has no guarantee that users will provide useful feedback. If providing feedback is made mandatory in order to use the system and this is coupled with a collection mechanism that users deem irksome, the quality of feedback becomes dubious as unwilling users "game" the feedback process to make it as unobtrusive as possible.

For these reasons, CDA mainly relies on implicit feedback, which is obtained during the course of normal application usage, and does not require users to perform any additional interaction other than what is required for completing the task. While CDA does not require explicit feedback, this can be exploited when available.

A problem with relying solely on implicit feedback is that due to the ambiguity inherent in implicit positive feedback, CDA might behave in a wasteful manner. For instance, CDA might not be able to gain sufficient preference information and provide higher-fidelity images than needed for a task. This issue can be addressed in various ways. CDA can probe for user preferences by over-adapting content (e.g., serving images at lower fidelity) so as to solicit implicit negative feedback until it accumulates sufficient history. In a production environment where the preferences of users change over time, this process would have to be repeated to keep the history up to date and ensure high quality predictions.

2.4. Practical Considerations

To be effective, CDA requires a critical mass of viewers of the same content object in order for communities to be identified. In addition, the client application must be able to obtain feedback from the user and relay it to the community center. If implicit feedback is utilized, it is also imperative that the application allow iteration in the presentation of the content and the content must be amenable to an iterative process.

Even in cases where significant reuse of content is not apparent, such as with dynamic content, CDA might prove useful. First, many dynamic web pages are built from a set of static objects (such as images). Here, the scenario is no different because the histories are maintained for individual objects that do no 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 a community will form. 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 dy-

namic content that bears no resemblance to past instances, then CDA will not apply.

Most of the complexity inherent to CDA is relegated to the resource-rich servers hosting the CDA community center proxies. Therefore, mobile clients are shielded from the resource intensive tasks associated with storing large history logs, running the prediction mechanisms, and performing transcoding. Moreover, logs can be truncated or characterized to save space.

User privacy may be a concern in CDA. Whereas many other systems can track the data that was accessed by a user, CDA 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 identifying communities, the size of history logs maintained and keeping track of individual user history versus aggregates, and how these aspects influence the quality of CDA's predictions is beyond the scope of this paper.

3. Case Study

To evaluate the effectiveness of CDA policies, we developed a case study that explores the problem of adapting image-rich web pages for browsing over a bandwidth-limited link. Specifically, we gathered traces of users as they performed a fixed set of tasks using adapted JPEG images on a laptop computer with a 56Kbps network connection. The fixed set of tasks groups the users into one highly-cohesive community. We choose a laptop (as opposed to a PDA or cell phone) for the client device, to simplify the system and eliminate any effects introduced by changes to page layout. We controlled the bandwidth between the laptop and the proxy by interposing a network emulator that simulates a 56Kbps wireless network link. The specific speed that we selected is optimistic when compared with wide area wireless network connectivity available to mobile devices and cellular phones in practice. Finally, the focus on one adaptation type (JPEG image compression) helps simplify evaluation.

The trace-gathering system initially presents users with low-fidelity versions of the web images, and keeps track of users as they refine the fidelity of images to complete a task. Our experiments are designed to motivate users to successfully complete the tasks without asking for unnecessary refinements. As such, the version at which users stop requesting improvements is the minimum fidelity that satisfies the user, and thus is termed the *optimal* fidelity. This is a good approximation to the real-world where network bandwidth is not free and users try to limit usage. Thus, the resulting traces capture the optimal fidelity required by users to complete a set of assigned tasks. The optimal fidelity for specific images may vary between users because people have varying preferences. In Section 4, we use these traces to evaluate the quality of predictions made by CDA and rule-based policies.

While the trace gathering system shares many ideas with CDA (users can iteratively change adaptation decisions and these actions are logged), the system is not a working CDA prototype as it does not do any prediction or use history of previous accesses. We decided to build a trace-gathering system rather than a prototype of a CDA system because we wanted to experiment with a large number of adaptation policies. Moreover, carrying out analysis on live and external web content requires extreme care because content updates and the vagaries of the network can potentially make valid comparisons of user experiences impossible.

In this section we first describe the trace-gathering system and then describe the tasks that we asked the users in our experiments to perform. Finally, we describe the setup and methodology of our case study.

3.1. Implementation

The trace-gathering system allows users to browse static web content over a bandwidth-limited network link, and request fidelity improvements to images. The trace-gathering system consists of three main components: an adaptation-friendly web browser, a transcoding/logging proxy, and a web server. The web browser runs on a laptop with a 14-inch screen while the web server and the transcoding/logging proxy run on desktop computers. The proxy is interposed in the network path between the laptop and the web server on a closed network in our lab. To control bandwidth, a network simulator is placed between the laptop and the rest of the system.

We implemented our adaptation-friendly web browser by extending Microsoft Internet Explorer 6 (IE6) with an ActiveX plug-in [6] that allows users to request fidelity refinements for images by right-clicking on an image. The proxy converts images embedded in HTML documents obtained from the web server into a progressive JPEG representation[1] [16]. The progressive JPEG images are then divided into pieces of equal file size, which can be served to clients on demand. To create traces of user interaction with adapted data, the proxy logs all user requests for fidelity refinements.

The system operates as follows. When loading an HTML page, the proxy parses the HTML to extract all image references, downloads the images, converts them to progressive JPEG, and slices them into equally-sized pieces (10 slices for regular images and 3 slices for thumbnail images). The slices map directly to fidelity levels and thus regular images have 10 fidelities while thumbnails have 3. Upon first loading the page, IE6 renders the images with the lowest fidelity (i.e., the

1 A useful property of a progressive image format, such as progressive JPEG, is that any prefix of the file for an image results in a complete, albeit lower quality, rendering of the image. As the prefix increases in length, the image quality approaches its maximum.

first progressive JPEG slice). The user, however, will immediately recognize that the images have been adapted because all the loaded images are presented with very low fidelity. The user can then request an improvement in the fidelity of a specific image by right clicking on it. The right-click event is trapped by the ActiveX plug-in which instructs IE6 to request additional data that would allow the specific image to be presented with a higher fidelity level.

3.2. Web Sites and Tasks

An ideal web site to be used in the study would involve a large number of images to maximize the opportunities for adaptation predictions. Therefore, we developed three image-rich web sites: a car show, an electronic store (e-store), and a map service. All the web sites, including all image content, are static and unchanging.

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). Unfortunately, such communities of interest cannot form spontaneously in a lab experiment with a limited number of participants which requires us to use specific tasks to compensate for this limitation. In the car show and e-store sites, we motivate similar behavior by specifying tasks where users have to look for a specific feature, whereas in the map service we specify a goal-based task. The tasks are such that not all images are relevant to their completion, and are designed to provide a common objective to users in order to create a community. By giving users a common and explicit set of tasks involving the viewing of images, we effectively force them into a single community. Therefore, this is the idealized case for CDA from the perspective that we are able to identify a community composed of members with a singleness of purpose.

3.2.1. Car Show The car show web site allows users to browse 6 images of cars in sequential order. 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.

This task is feature-driven because users are specifically asked to look for cars with license plates. We expect that the task will result in locality because it is possible to determine that 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.

3.2.2. E-store The second web site is an electronic store (e-store) which allows users to shop online for electronic products. It presents the users with four product categories: PDAs, cell phones, MP3 players and AIBO robots. Accessories are also appended at the end of each category. There are two sets of images associated with each product: a thumbnail and a full-size snapshot. Thumbnails are loaded with the product by default, whereas full-size snapshots are loaded upon selection by the user. Users are asked to purchase a PDA with a red mark at the bottom right, a cellular phone with a label that reads "Sierra Wireless", and an AIBO robot wearing a blue and silver necklace.

Similar to the car show, this task is also feature-driven and exhibits locality because users have to find a specific product in each product category. It differs from the car show in that users are not forced to browse through every image, thus providing different user behavior; for example, users may choose to browse through all PDAs in order, or randomly select a few until they find the desired one. The specific task of finding a product with a particular feature serves to artificially group users into a community.

3.2.3. Map Service The last web site is a map service which allows users to view a detailed online map of the University of Toronto. The map is actually made of an 8 X 8 grid of images. Upon loading the web site, all images in the grid are visible at low fidelity. It is possible to refine individual images in the 8 X 8 grid by right-clicking on them. Users are asked to locate a path between the main University of Toronto library and a popular subway station, and record the names of the buildings along the path.

Unlike previous tasks in the e-store and car show which are feature-driven, this task is goal-driven because the intent is to find directions to travel from a source to a destination. Images of the sliced map that are on the path between the source and destination location, need to be increased in fidelity in order to record the building names. Locality in this case occurs because users are familiar with the map.

3.3. Methodology and Setup

A total of 28 users participated in the study. All users are Computer Science undergraduate students at the University of Toronto. Each user is asked to sit in front of our client laptop and perform specific tasks on the three web sites described in Section 3.2. On-line instructions were presented to the users explaining that they were accessing content over a bandwidth-limited link, that an adaptive proxy would transcode images into low-fidelity versions to reduce download time, and that they could improve image fidelity to perform the task. The incentives in our case study are structured so that users attempt to complete the tasks with the fewest requests for refinement, which mimics real-life where users have a monetary motivation for saving bandwidth.

4. Experimental Evaluation

In this section, we first describe the CDA and rule-based policies that we use in our experiments. We then compare the effectiveness with which the predictions generated by the rule-based and CDA policies match the fidelity levels captured in our user traces. Experimental results show that CDA policies outperform rule-based policies, consuming less bandwidth and requiring users to click fewer times. Finally,

we present an analysis of the user-trace data obtained from our case study. Our analysis shows that most users select similar fidelities, and that image fidelity is dependent on the data's semantics and its relationship to the tasks.

4.1. Policies

We consider 5 rule-based policies: FIXED1, FIXED2, FIXED4, FIXED10 and SIZE2. The FIXED*i* policies serve all images at a fixed fidelity level *i*. SIZE2 groups all images into 2 size-based classes with each class having the same number of images. Since we have 100 images in total, we place the smallest 50 images into the first class, while the remainder are placed into the other. Images with larger sizes are served at fidelity 3 while the smaller images are served at fidelity 7 (fidelities 1 and 3 for thumbnails, respectively). This policy captures the intuition that larger images can be effectively viewed at lower fidelity.

FIXED1 and FIXED10 provide upper bounds for comparison with other policies. FIXED1 serves all images at the lowest fidelity. FIXED1 does not waste data, but results in the maximum number of user requests for fidelity refinements. In contrast, FIXED10 serves all images at their highest fidelity. FIXED10 does not require users to request fidelity refinements, but wastes the maximum amount of data.

We consider 8 CDA policies: MAX, MAX2, AVG, AVG2, AVG3, MEDIAN, MODE, and UPPER60. When no history is available for an image, the policies select the lowest fidelity. Otherwise, the policies select the maximum, average, median, and mode of the entries in the history log, respectively. The policies ending with a 2 or 3 implement the operation indicated by the policy's name but limit the scope of the history log to just the last 2 or 3 entries. For example, AVG2 and AVG3 select the average fidelity chosen by the last 2 and 3 users in the history log, respectively. The restricted history makes the policies more resilient to distortion because every user remains in history for a limited amount of time, after which their fidelity selection has no impact.

The UPPER60 policy constructs a probability distribution function of user requests based on history and selects the fidelity at which 60% of the user requests get covered. For example, out of 10 users, if 2 of them select fidelity 2, 6 of them select fidelity 3 and 2 of them select fidelity 4, UPPER60 selects fidelity 3.

4.2. CDA Performance

The traces gathered in our user study capture the fidelity level that each user considered optimal for each image. We next evaluate the extent to which the predictions generated by the rule-based and CDA policies match the fidelity levels captured in the user traces.

We evaluate policies using two performance metrics: *wastage* (wasted bandwidth) and *extra clicks* required by the user. These metrics capture the tradeoff in adapting content between conserving resources and inconveniencing users when they are forced to interact with the system. For every user request for an image, we apply each of the policies to predict the fidelity at which to serve that image. If a policy selects a fidelity that is higher than the optimal (*overshoot*), we have wastage amounting to the difference between the size of the data that the user got served and that which corresponds to their desired fidelity level. If a policy selects a fidelity lower than the optimal (*undershoot*), the user is forced to provide feedback to the system in order to reach the optimal fidelity. In this case, we have extra clicks amounting to the difference between the optimal fidelity and that selected by the policy.

For CDA, our goal is to study the situation where the system is out of its initial transitionary state. Since our study only consisted of 28 participants, when measuring the performance

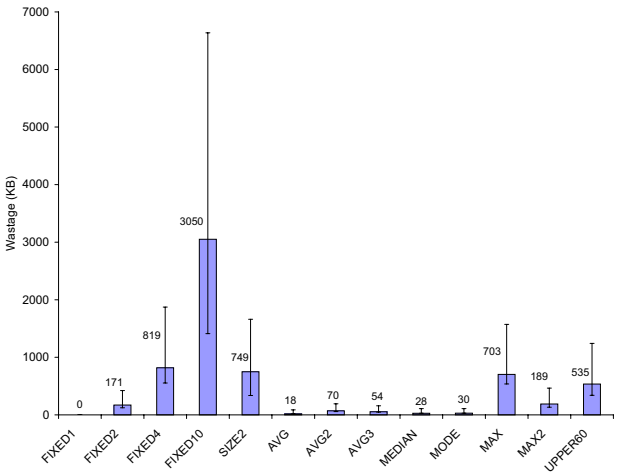

Figure 2. Wastage for various adaptation policies.

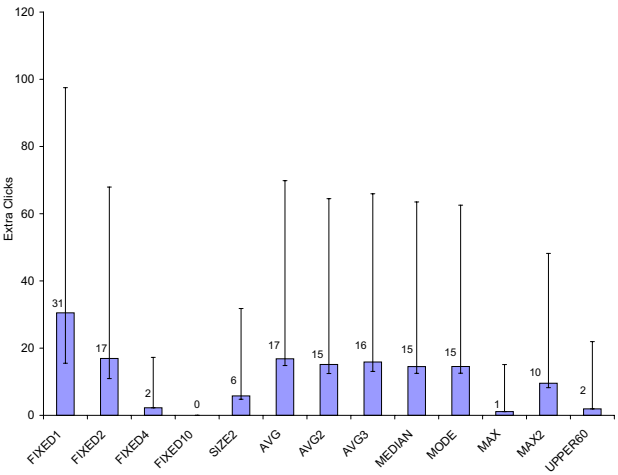

Figure 3. Extra clicks for various adaptation policies.

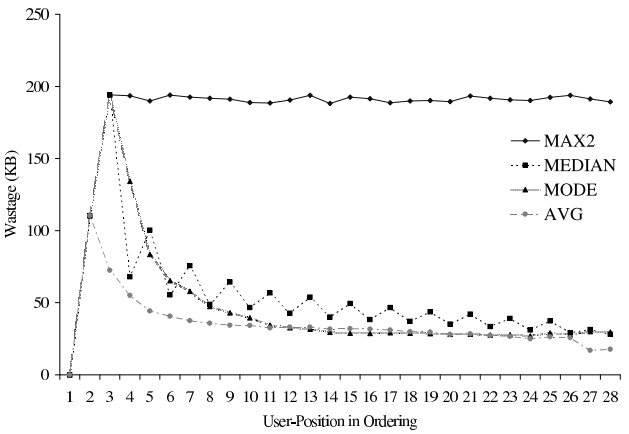

Figure 4. Wastage convergence for various adaptation policies.

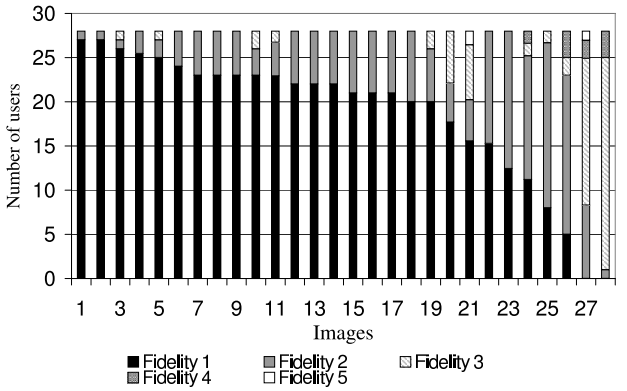

Figure 5. E-store: Fidelity distribution for users.

of each user, we rearrange the trace to make it seem that they accessed content after all the other users in the trace. Because the predictions made by some CDA policies are based not only on the fidelity selected by previous users in the trace, but also on the ordering of these accesses, we consider different permutations of user orderings. Thus, we fix each of the 28 users to be the last one in the ordering, and then generate a hundred random orderings of the other 27 users. The performance of the user is an average of the 100 runs. When reporting the performance of each of the CDA policies, we average the results of the 28 users (an average over 2800 trials). Given that the predictions generated by rule-based policies do not depend on history, no reordering or multiple runs were necessary in computing the average performance experienced by the 28 users.

Figure 2 shows the average amount of data wasted and Figure 3 shows the average number of clicks required by each policy over all three web sites. The bars indicate the range of the respective metric over the entire set of users. The size of all images at their full fidelity amounts to just over 5 MB. Users consumed an average of 720 KB of data to perform all tasks. FIXED10, which serves images at the highest fidelity, wastes close to 3 MB incurring over 325% overhead. The wastage for FIXED10 is less than the full size of the dataset would suggest (5 MB - 720 KB) because not all images are loaded by all users.

We observe the following when we compare the performance of rule-based and CDA policies. First, where a rule-based policy and a CDA policy produce the same wastage, we find that the CDA policy requires fewer extra clicks. For example, the rule-based policy SIZE2 produces similar wastage to the CDA policy MAX (749KB versus 703KB), but the MAX policy requires 5 times less extra clicks than SIZE2 (1 click versus 6 clicks). Second, in cases where a rule-based policy and a CDA policy require the same number of extra clicks, the CDA policy produces less wastage. For example, the rule-based policy FIXED2 requires the same number of extra clicks as the CDA policy AVG (17 clicks), but the AVG policy has 90% less wastage than FIXED2 (18KB versus 171KB). Third, there are cases where CDA policies require both fewer extra clicks and lower wastage compared to rule-based policies. For example, the CDA policies AVG2, AVG3, MEDIAN, and MODE have 60%, 68%, 84%, and 82% less wastage and 12%, 6%, 12%, and 12% less extra clicks than FIXED2, respectively.

Figure 4 shows the rapid convergence of wastage for the MAX2, MED, MODE and AVG CDA policies. The values plotted are averages over 2800 trials with different user-orderings. The policies do not behave optimally for the first few users but the wastage quickly reduces because better predictions are made as more history accumulates. Therefore, we conclude that CDA policies do not require a large number of previous user accesses to behave well.

4.3. Analysis of User Traces

An inspection of the traces showed that while the three web sites (car show, e-store and map service) have a combined total of 100 images, not all are loaded by every user, and that a fidelity level of 4 was sufficient for most users to complete the tasks. A small number of images were requested by one or two users at fidelity levels higher than 5. Given that we have no control over awkward user behavior, it is reasonable to consider these few points as random noise.

In our case study, locality occurs when users select similar optimal fidelity levels for an image, with each fidelity level constituting a distinct *fidelity class* for the image. We find significant locality in the optimal fidelity of users in our experiments. Out of the 77 images that are loaded, 32 images have only one fidelity class, which in most cases the fidelity level is 1. Of the 45 images that have multiple fidelity classes, 20 images have two fidelity classes, while the remainder have two adjacent fidelity classes that cover over 80% of the fidelity selections. For all images, one fidelity class covers at least 43% of the fidelity selections.

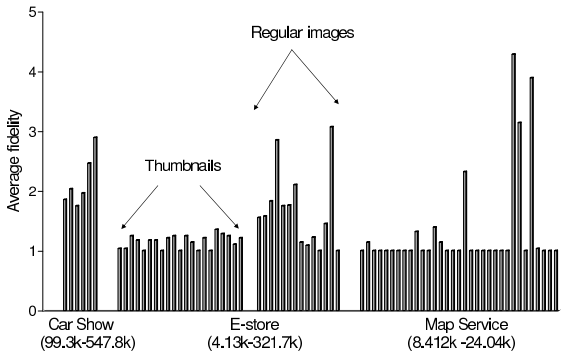

Figure 6. Average user requested fidelity.

Figure 5 exemplifies these trends. The figure shows the breakdown of fidelity classes for images in the e-store web site that had two or more fidelity classes. For a given image, each pattern in the image bar shows the number of users that requested a particular fidelity class. The images are sorted in decreasing order by number of requests for fidelity class 1 followed by requests for fidelity class 2.

We find that images that are critical to the task show a large number of fidelity classes, indicating that a large diversity in selected fidelity classes is a good predictor of image relevance to a specific task. For example, in the e-store, the last three bars in Figure 5 represent products that users were told to purchase. Similarly, the images that show a lot of diversity in the map service are the ones on the path between the source and destination (not shown). Despite this diversity, the majority of requests fall in two adjacent fidelity classes.

Figure 6 plots the average fidelity level selected for all images (sorted by size and grouped by web site and purpose) and illustrates the lack of strong correlation between fidelity level and image size, purpose (thumbnail vs. regular image), or location (across web sites). An ANOVA test at an $\alpha = 0.05$ significance level confirmed that the mean image fidelity is not the same for the three tasks. Also, we computed a Pearson's correlation coefficient value of 0.195 between image file size and desired fidelity, indicating a very weak positive linear relationship. This suggests that rule-based or constraint-based approaches to image adaptation will fail to provide an optimal browsing experience.

5. Related Work

There is significant research on content adaptation for mobile devices [4, 6, 10, 11, 17–20, 24, 25], and even a few commercial adaptation systems have been deployed [4, 15].

Two main techniques for automatic adaptation policy generation are rule-based [4, 14, 24–26] and constraint-based [7, 18, 19, 26] adaptation. In both approaches, adaptation policies are defined using high-level programming languages or mathematical formulas [7, 14]. Rule-based systems rely on high-level rules to guide the adaptation process. When adapting an object, the system determines the subset of rules that apply and adapts accordingly (e.g., convert images larger than 50 KB to progressive JPEG images). Constraint-based adaptation extends rule-based adaptation to encode tradeoffs between possible adaptation strategies. A constraint captures, in a mathematical formula, the relationship between resource consumption and user satisfaction for a specific adaptation. An automatic solver adapts content by finding a solution that meets all constraints, minimizes resource consumption, and maximizes user satisfaction. 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). Moreover, determining the relationship between user satisfaction and content metrics, such as resolution or frame rate, is hard and often depends on the semantics of the content being adapted and the user's task, which is rarely taken into consideration. In contrast, CDA generates content-specific adaptation policies that take into account the relevance of the data to the user's tasks.

Narayanan [19] uses history logs for application adaptation. By monitoring resource consumption, Narayanan can predict the application's operating mode that achieves a specified battery life and maximizes content fidelity. Narayanan's approach assumes that adaptation decisions are driven mainly by the applications and are largely independent of the input. In contrast, CDA can be used effectively with applications where the optimal adaptation strategy is highly dependent on the content being operated on.

CDA is related to previous efforts on recommendation-based systems. Most recommendation systems [2, 5, 13, 27] use collaborative filtering, in which people collaborate to help one another perform filtering by recording their reactions to documents they read. Balabanovic et al., [3] add the ability to evaluate and provide feedback in order to learn and improve on the recommendations. A collection of histories [21] can be created and then mined to recommend to the user a set of candidate functions and to detect users' erroneous behavior. Semantics can be used to build a model of the user [12] such as that used by online retailers like Amazon.com, which can then be used to recommend other items in the same class of products. CDA is a radical new use of the community-based recommendation concept – adaptation prediction. Whereas previous efforts have focused on predicting what content to obtain, CDA focuses on the question of how to adapt this content to offset the resource limitations of mobile devices.

6. Conclusions and Future Work

Community-Driven Adaptation (CDA) is an automatic content adaptation technique for pervasive-computing environments in which content is adapted based on user feedback and the semantics of the content pertaining to the partic-

ular task that the user is performing. We gathered traces of users browsing web sites with adapted images. The traces capture the minimum fidelity that specific users perceived as sufficient to carry out a specific task. Simulations on the traces show that for a highly-cohesive community where all the users perform the same tasks, CDA policies that learn from previous user accesses easily outperform rule-based policies that are blind to the content semantics. Compared to rule-based policies, CDA policies provide up to 90% reduction in wastage of network bandwidth and up to 50% reduction in user requests to improve fidelity.

In this paper, we artificially constrain users to a single community to demonstrate that the CDA technique works. In the future, we plan to research techniques for automatically grouping users into appropriate communities. In addition, we plan to evaluate the effectiveness of CDA for adapting other data types (such as video and audio), the applicability of CDA to other adaptation domains (such as page layout and content modality), and the extent to which specific user interfaces affect the quality of the user-provided feedback and the resulting adaptation prediction.

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] 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] 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.

[5] CiteSeer. `http://citeseer.ist.psu.edu/`.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

[12] R. Ghani and A. Fano. Building recommender systems using a knowledge base of product semantics. In *2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems*, Malaga, Spain, May 2002.

[13] 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.

[14] 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.

[15] iAnywhere Solutions. Avantgo. `www.avantgo.com`.

[16] Independent JPEG Group. `www.ijg.org`.

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

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

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: a system for sharing recommendations. *Commun. ACM*, 40(3):59–62, 1997.

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