Efficient and Transparent Dynamic Content Updates for Mobile Clients

Trevor Armstrong
Department of Electrical and Computer Engineering
University of Toronto, Canada
trevor@eecg.toronto.edu

Olivier Trescases
Department of Electrical and Computer Engineering
University of Toronto, Canada
trescas@vrg.toronto.edu

Cristiana Amza
Department of Electrical and Computer Engineering
University of Toronto, Canada
amza@eecg.toronto.edu

Eyal de Lara
Department of Computer Science
University of Toronto, Canada
delara@cs.toronto.edu

ABSTRACT

We introduce a novel infrastructure supporting automatic updates for dynamic content browsing on resource constrained mobile devices. Currently, the client is forced to continuously poll for updates from potentially different data sources, such as, e-commerce, on-line auctions, stock and weather sites, to stay up to date with potential changes in content. We employ a pair of proxies, located on the mobile client and on a fully-connected edge server, respectively, to minimize the battery consumption caused by wireless data transfers to and from the mobile device. The client specifies her interest in changes to specific parts of pages by highlighting portions of already loaded web pages in her browser. The edge proxy polls the web servers involved, and if relevant changes have occurred, it aggregates the updates as one batch to be sent to the client. The proxy running on the mobile device can pull these updates from the edge proxy, either on-demand or periodically, or can listen for pushed updates initiated by the edge proxy. We also use SMS messages to indicate available updates and to inform the user of which pages have changed. Our approach is fully implemented using two alternative wireless networking technologies, 802.11 and GPRS. Furthermore, we leverage our SMS feature to implement and evaluate a hybrid approach which chooses either 802.11 or GPRS depending on the size of the update batch. Our evaluation explores the data transfer savings enabled by our proxy-based infrastructure and the energy consumption when using each of the two networking capabilities and the hybrid approach. Our results show that our proxy system saves data transfers to and from the mobile device by an order of magnitude and battery consumption by up to a factor of 4.5, compared to the client-initiated continuous polling approach. Our results also show that the batching effect of our proxy reduces energy consumption even in the case where the user never visits the same page twice.

MobiSys'06, June 19–22, 2006, Uppsala, Sweden.

Categories and Subject Descriptors

C.5.3 [**Computer System Implementation**]: Microcomputers—*portable devices*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.8 [**Operating Systems**]: Performance—*measurements*

General Terms

management, measurement, experimentation

Keywords

power management, mobile wireless communication, proxy, caching, prefetching, batching, energy measurement

1. INTRODUCTION

In this paper, we introduce an automated and efficient approach for browsing HTML pages with dynamically changing content on mobile devices. Following the fluctuations of the favorite currency, stock value, or auction currently requires the user to reload all the pages in order to capture any changes to the data. The costs of these data transfers to the user come in many forms, including slow data access, excessive battery consumption on the device and inconvenience due to the user's active involvement in constant data reload.

We observe that, while web pages may change frequently due to systemic reasons, such as updating the time of day or an add banner, the user relevant content does not change much. Based on this key observation, we introduce a general purpose infrastructure that allows the browsed HTML pages to be seamlessly updated only when content of interest to the user changes. Our approach greatly reduces the costs of updates by: i) allowing the users to mark the parts of each page that are of interest to them, ii) offloading the task of

determining when those parts have changed to a resource-rich proxy and iii) leveraging the proxy for batching those updates and sending them to the user's device periodically.

We expect that our system will be useful in two kinds of browsing situations: Our first target is providing seamless low-cost content updates during active client web browsing. Imagine a user browsing dynamic content on her PDA during her daily commute or at an airport terminal waiting for her flight. We leverage our resource-rich proxy to save data transfers for both the case where the user wants to keep up to date with rapidly changing content for her favorite pages as well as for the case of browsing to random pages.

Our second target scenario is automatic periodic content refresh for the user's favorite content, for subsequent browsing while disconnected. This scenario corresponds to a user carrying a handheld device in her pocket, and having her preferred content (news, weather, stocks, etc) automatically updated whenever her device has the opportunity to network and significant changes to the content have occurred.

In our system, dynamic content is cached both within a local mobile client proxy and on an edge proxy with an always-on high-bandwidth low-latency connection to the Internet. The client registers her interest with the edge proxy. The edge proxy keeps track of the web pages cached at each client and polls the web servers involved for changes. We allow the mobile client to specify their interest in regions within each web page, using a very simple interface, by highlighting portions of already loaded web pages. We also refine our interface by allowing the user to specify sensitivity thresholds for changes to numerical values. Finally, this specified interest allows the edge proxy to selectively accumulate content updates for all pages in the client's profile, and propagate updates as a single batch only when changes of interest to the client occur.

We deployed an actual proxy in our lab from which our mobile device can connect using two alternative wireless networking capabilities: 802.11 and cellular communication over GPRS. Each of these networking capabilities offer different trade-offs in terms of data download costs. Specifically, access to content over cellular networks is ubiquitous and low power, but is relatively slow. On the other hand, transfers over WiFi (802.11) are fast, but have high energy costs. Indeed, an 802.11 card can reduce the battery lifetime of a PDA by up to a factor of six when in continuously active mode and by a factor of nearly two when in power saving mode [1].

In our experiments we measure the data transfer and energy savings for several dynamic content refresh schemes. Specifically, we implement and compare a poll-based scheme, where the mobile proxy periodically polls the edge proxy for updates, and a push-based approach, where the edge proxy pushes updates to the device based on a schedule. We implement and measure the poll-based and push-based proxies and compare against a baseline without proxies on both WiFi and GPRS. Finally, we leverage lightweight SMS text messaging to signal when new updates have been created. The edge proxy creates a text message for the client, on which it piggybacks information such as the number of updates, size of updates, and pages that have changed. Thus, the mobile proxy can use this information to make an intelligent decision on whether to use its WiFi or cellular connection to download the updates. Specifically, the energy expenditure of transferring a short message on GPRS is expected to be dwarfed by the energy overhead of turning on the WiFi card. Conversely, for large messages, the superior download speed of WiFi translates into overall energy savings even if transfers occur at a higher power level.

We present results from experiments conducted by replaying four 3-hour URL traces collected from the following live sites: EBay.ca [2], the CNN Toronto weather page [3], the currency web site XE.com [4], and the financial site Yahoo! Finance [5]. We select parts of these web pages that the user would normally be interested in, such as, a particular auction's current bid, the current temperature, the value of a particular currency, and the value of a particular bond, respectively.

Our results show that, in all cases, combining the update filtering with the batching edge proxy results in significant cost reductions. The client can learn in a single message exchange that there are no updates for any of the pages of interest. Conversely, when significant changes occur, transferring all relevant updates in a single batch bypasses much of the latency involved in WWAN web browsing, hence conserving battery power. We show that our proxy-based infrastructure reduces data transfers and energy consumption when using either of the two communication capabilities, i.e. GPRS and WiFi, alone. Furthermore, we have determined the threshold message size, for which using WiFi is more energy efficient than GPRS. This information is leveraged in our hybrid approach which uses both wireless connections in conjunction with SMS messages and allows for even further energy savings. Overall, we save data transfers to and from the mobile device by an order of magnitude and battery consumption by up to a factor of 4.5, compared to the client-initiated proxyless approach. Our results also show that even when the client is browsing new, previously unvisited, web pages, our proxy system has a beneficial prefetching effect by batching downloads for the page and all its embedded objects into one data transfer. As a result, in this case our proxy reduces energy expenditure by 69% when browsing over GPRS, and 15% when browsing over WiFi.

The outline of the rest of the paper is as follows. Section 2 introduces our proxy based approach. Section 3 presents our experimental platform and methodology. Section 4 presents our results. We provide a summary of our results, and a discussion of our contributions in Section 5. Section 6 discusses related work. Section 7 concludes the paper.

2. OUR PROXY FRAMEWORK

We assume a scenario where a mobile user is interested in browsing dynamic content from various web servers. The client interaction with many of today's web servers is repetitive in nature, such as, constantly polling an EBay auction to check the status of a bid, or refreshing a page that contains stock quotes to track the changing values of a stock. While browser caches support "get if modified since" mechanisms, this typically fails to save any data transfers due to frequent updates to parts of the page that are largely irrelevant to the user. These changes include ad banners or the time of day, and although the user may not be interested in them, they usually result in the page being reloaded almost every time.

To address this problem, we provide a simple mechanism that allows users to specify the information they want to keep track of in general HTML pages. Based on the user interest, we offload the polling work from the mobile device

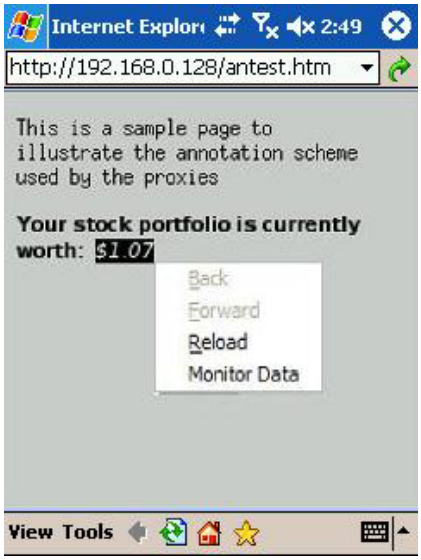


Figure 1: Screen Shot for Illustrating Client Interface

to an edge proxy connected to the Internet over a high capacity link. The proxy is a regular computer with no power limitations and sufficient memory. The edge proxy performs high-frequency polling actions for detecting updates to the content of interest to the mobile clients.

We describe the interface that we offer the mobile client and the way we keep track of the client's interest in section 2.1. We then describe the components of our proxy architecture and its basic operation in section 2.2 and section 2.3, respectively. Building on this basic framework, section 2.4 introduces our proxy enhancements for communication and energy savings.

2.1 Client Interface

The user specifies her interest in changes to specific parts of each page by highlighting portions of the web page on her device screen, as illustrated in Figure 1. The end points of a highlighted region serve as the start and end points of an annotation that the system captures.

To keep track of the mobile client's interest in specific page regions even while the content changes, we use a well-documented tree technique for maintaining robust HTML document locations [6]. This technique has been shown to robustly keep track of a location within a web document, in the face of typical value changes to dynamic content and even in the case of structural changes to the document, such as paragraph reordering or deletion.

Tree walks are the central component of robust HTML locations. A tree walk describes the path from the root of the document, through internal structural nodes, to a point with plain content at a leaf. Since tree walks incrementally refine the structural position in the document as the walk proceeds from root to leaf, they are robust for dynamic content page changes, such as with stock quotes, where the content itself changes while the structural position remains constant. In many cases, even when structural changes have occurred, it is possible to find the content of interest by visiting sibling nodes in the tree and using additional context information.

Finally, our interface also allows the client to specify thresholds of meaningful change for numerical values.

2.2 System Architecture Components

Figure 2 shows the two main components of our system: the mobile device proxy and the edge server proxy. The mobile proxy resides on the mobile device. It consists of a proxy that intercepts client web requests, a cache for storing the responses to previous requests, and a hardware manager which controls the state of the wireless connections available on the device. The mobile proxy's main job is to communicate with the edge server proxy and process any cache updates. The hardware manager on the mobile device is responsible for determining which wireless interface the inter-proxy communication should use. The hardware manager makes its decision based on user defined preferences. The user can choose to prefer GPRS-only, WiFi-only or an adaptive GPRS/WiFi hybrid with the goal of optimizing energy consumption automatically. For example, in an interactive scenario of actual user browsing, where download speed is important or if avoiding monetary costs is paramount to the user, the user would set a preference for the 802.11 connection. In this case, the hardware manager first checks for the availability of an 802.11 access point before falling back to the GPRS connection if one is not available. In the hybrid case, the hardware manager bases its decision on which interface to use on the size of the data to be transferred. A long download of a large update on GPRS may consume more energy overall than the equivalent transfer over 802.11, even if the GPRS connection uses relatively less power.

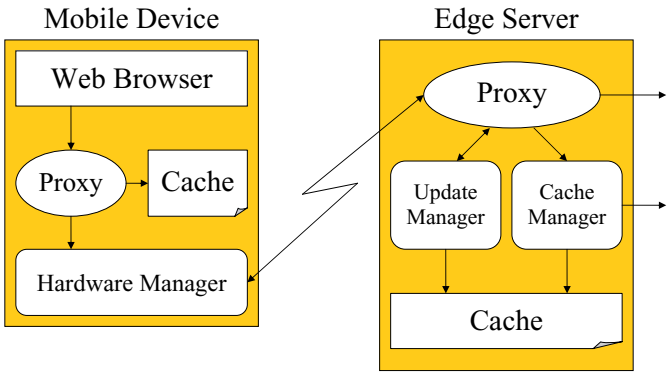


Figure 2: System Architecture

The edge server proxy is placed on any well connected computer. The edge server proxy consists of four components: proxy process, cache manager, cache, and update manager. The proxy process is an event driven server which interacts with multiple clients and serves their requests either from the cache or by directly connecting to the web server(s) in question. The cache manager consists of an interface to the cache and a thread pool. The cache manager's responsibility is to keep the cache up to date. Each thread periodically polls the web servers that a particular cache entry references, checking for any changes. The cache stores the interest profiles for all the mobile devices that registered their interest with the edge proxy. When a cached

page is changed, the update manager adds a reference to the changed content to the update batch of each mobile device that has registered interest in that particular page.

2.3 Operation

When a mobile device first joins the system, it registers with the edge server proxy. The edge server proxy assigns each device a unique id so that it can subsequently differentiate between devices in the system. Differentiating based on IP address is not a sufficient means, since a mobile device may change IP addresses several times each day.

When a request is issued by the web browser on the device, the mobile proxy checks its cache. If the cache contains the corresponding response (local cache hit), the response is returned immediately to the web browser and no wireless communication occurs. If the response is not found in the cache (local cache miss), the mobile proxy forwards the request to the edge server proxy. The edge server proxy, in turn, checks its cache for the response and returns it from its cache if it is there. Otherwise the request is forwarded to the actual web server. If the response is a HTML page, the edge server proxy prefetches all the embedded objects within that page and batches them with the response to be delivered to the mobile client in one transmission. Any pending cache updates are also included in the batch transfer.

Upon receiving the response from the edge server, the mobile proxy caches the response and updates its cache with any other additional files included in the transfer. The response is then returned to the web browser. The client proxy acknowledges the receipt of any updates, such that the edge server proxy can remove those updates from the update manager's list for that device.

In our system, the mobile proxy learns that cache updates are available through three alternative means:

- Polling the edge server.
- Receiving pushed updates.
- Receiving a SMS message from the edge server.

In the polling based scheme, the mobile proxy periodically polls the edge server proxy asking whether any updates are available. This periodic content refresh occurs automatically during active browsing sessions in order to keep the local client cache up to date, and in turn to minimize client perceived staleness and waiting time.

Alternatively, for the push based approach, the mobile proxy listens on a particular port for incoming updates initiated by the edge server proxy. In this situation, the edge server proxy requires a valid IP address for the client. Hence, when the device's IP address changes, the mobile proxy is required to contact the edge server proxy to give it its new address.

The previous two methods require the user to have an active data connection in order to learn of updates. However, there may be times when the user is in a disconnected state[1] and would still like to receive notifications about changes to pages of interest. By harnessing the existing Short Message Service (SMS) infrastructure that most wireless carriers provide, our proxy system is able to provide this functionality.

In this scenario, when there are updates for the mobile proxy, the edge server proxy constructs and sends a single SMS message that is divided into two parts. The first part consists of control information for the mobile proxy, including the number of updates and the size of the download. The second part is an update summary, which is intended for the user. This summary includes a list of the pages that have changed, and if particular values were being monitored, the changes that occurred[2].

The mobile proxy intercepts any incoming SMS messages from the edge proxy. It strips off the control portion of the message and passes the remaining user portion back to the SMS handler for delivery to the user. As a result, both the user and the mobile proxy have information describing the updates. The mobile proxy uses this information towards making an intelligent decision on the appropriate connection to use for acquiring the updates. As a positive side effect, the SMS notification may enable the user to avoid using the browser altogether. For example, if the user is interested in a stock quote, then the SMS message containing the new value may convey all the information required. A similar SMS notification feature is provided by some vendors for certain proprietary data. However our system allows the user to use this functionality on virtually any data on the Internet.

2.4 Infrastructure Enhancements for Communication and Energy Savings

The edge proxy keeps track of each client's interest in portions of HTML pages, expressed through the interface described in Section 2.1, as a client profile. The edge proxy polls the respective web servers for changes to the web pages in each client's profile. In order to account for the effects of java script in the HTML file, the edge proxy renders each page locally and uses the rendered source for comparison. If the pages have been modified, the proxy determines whether these changes are meaningful for the respective client. Specifically, the edge proxy determines whether the differences between the old and new versions of a page are indeed located within portions of the page of interest to the client. Furthermore, the edge proxy determines whether changes to numerical values that have a client-specified change threshold, exceed the recorded threshold. If a meaningful change occurs, the edge proxy accumulates the new version of the page as an update to be sent to the respective mobile client. As before, the edge proxy aggregates all updates to be sent to each client as a single update batch. Since each update overwrites the whole page, the number of updates in the update batch never exceeds the number of pages that the user has registered interest in.

3. EXPERIMENTAL METHODOLOGY

For our evaluation, we use a desktop PC running Redhat 9 as our edge proxy. This system contains dual Athlon 2600+ processors, 512 MB RAM, and a 100 Mbit/s Ethernet network connection. In addition, we equip this system with a second network interface card that interfaces the edge server proxy to a wireless access point.

The mobile device we use in the experiments is a HP iPAQ 6325. It runs the Windows Mobile 2003 operating system and comes equipped with a 168 MHz processor, 64 MB of ROM, and 64 MB of RAM. This device has built in 802.11b and GSM/GPRS interfaces as well.

[1]We assume that there is no valid data connection, but the device's cell phone is still on

[2]The size of the user section is restricted by the maximum total size of the message (160 characters for Rogers Wireless)

In order to facilitate repeatable experiments, we took 3 hour traces from 4 real web sites and we compare all our configurations by replaying these traces in real time (i.e., each experiment is a 3 hour experiment). These traces are served by an Apache web server running on the same desktop machine as the edge proxy.

3.1 Real World Traces

For the traces, we used four popular sites; EBay.ca [2](we choose an auction that would be ending near the end of the trace period), the CNN Weather page for Toronto [3], the currency site XE.com [4], and the Yahoo! Finance page [5].

To gather the traces, we created a Firefox browser extension that repeatedly visited each of the sites in the trace and saved each downloaded web page to disk. Each iteration was roughly 20 seconds after the previous one, resulting in approximately 733 copies of each site in the trace.

The content of the traced pages proved to be very dynamic. Changes in content were either due to the volatile nature of the information being presented (stock quotes or currency values), or due to the inclusion of random advertisements. The EBay auction is a good example of frequent minor changes to the page, while the main bid-related information itself changes relatively infrequently. The auction that was chosen had 18 bids at the start of the trace. By the end of the trace, 24 bids had been placed. However, by analyzing the trace files, we were able to determine that the auction page had changed 377 times. This is due to the fact that the page contains a value showing how much time is left in the auction. This value was reported at the granularity of seconds during the last hour of the auction, and minutes prior to that.

The other sites in the trace were even more volatile, changing upon every access. The Yahoo financial site contains a large amount of rapidly changing information, including the Dow, Nasdaq and NYSE, as well as several currency values. In addition, the site has several random advertisements. All these factors combine to cause the constant change. We monitored the value of the 10 year bond. This value changed 211 times over the 3 hour trace.

Similarly, the volatile nature of the currency values reported on XE.com resulted in a change to this page every time as well. We monitored the value of the Euro, which over the course of the 3 hour trace changed 365 times. This high rate of change is mainly due to the fact that XE.com reports the value to the nearest $0.00001.

The large number of random ads, along with a time of day value, on the CNN weather page were the main culprits for the numerous changes recorded for this page. This site changed on every access as well, while the actual temperature value reported changed only 3 times over the course of the trace.

3.2 Proxy-based and Standalone Configurations Used for Comparison

In the following section, we describe in detail the various proxy-based and standalone configurations we use for comparison with our main approach. By gradually introducing some of the features of our main proxy approach, we are able to demonstrate what aspects contribute to the overall wireless communication savings.

3.2.1 Baseline Configuration without Proxy

In our baseline configuration, the browser running on the mobile device polls all web sites periodically for the pages opened by the client for any change in the content. No proxies are used in this configuration. However, the browser's cache is fully functional.

3.2.2 Simple Proxy

In this configuration, we run the two proxies, the mobile device proxy and the edge proxy, and we use the edge proxy to poll for any changes to the data occurring at each separate data source. The proxy schedules an update to be sent to the client when there is *any* change to a web page. The edge proxy aggregates all updates to be sent to the user as described in our main algorithm. The mobile proxy pulls updates both upon a cache miss and periodically with the same interval as that of polling in the baseline configuration.

3.2.3 Intelligent Proxy

The intelligent proxy configuration is our proxy-based approach which filters out any updates to the mobile device if the parts of the page that the client is interested in have not changed. The client specifies interest by highlighting page regions through the interface described in section 2.1.

3.2.4 Thresholds Proxy

The thresholds proxy is an enhanced intelligent proxy where the client specifies her regions of interest within a web page, but can also specify a threshold of significant change for each numerical value. All updates for numerical value changes that are below the significant change threshold are filtered out by the edge proxy.

We use both a polling based and push-based thresholds proxy in our experiments. One drawback of our experimental setup is that our edge server is operating outside the Rogers GPRS network. As a result, our edge server is unable to create a connection to the device over GPRS as all incoming communication from an external source is blocked by the Rogers firewall. In order to facilitate push-based experiments over GPRS, our mobile proxy creates a persistent TCP connection with the edge server. Updates are then pushed to the mobile device over this connection.

3.3 Parameters used in Each Configuration

We use Internet Explorer (IE) as our web browser on the mobile device. However, we use a simple wrapper around it to mimic the user and drive the experiments. All communication uses HTTP/1.1. In our baseline configuration, IE is running alone on the mobile device. The web browser contains a cache of its own, and as a result, after the first round of communication, the majority of the requests consist of if-modified-since requests from the browser for validating the cached items.

The browser is set up to visit the four sites in the trace, once every 4 minutes. This means that over the 3 hour experiment, each of the 4 websites in the trace is loaded 45 times. The period with which the pages are loaded is irrelevant, except for allowing the experiment to complete in a reasonable amount of time and to allow for full download of the respective pages over WiFi or GPRS.

In our experiments, we select the data of interest as follows: the current bid for the E-Bay auction, the current temperature for the weather site, the value of the 10 year

bond for the financial site, and the current value of the Euro in US dollars for the currency site.

As reported in section 3.1, the values of the Euro and the 10 year bond are very volatile. This is mainly due to the resolution with which these sites present the values, the nearest \$0.00001 and 0.001%, respectively. For our threshold proxy configuration, we set conservative thresholds for each of these values in an attempt to observe the impact of threshold filtering on the results. We set the thresholds to a \$0.001 change for the Euro and a 0.005% change for the 10 year bond. We believe these settings give us a conservative estimate in our evaluation for the potential communication savings in most real scenarios.

3.4 Experimental Setup for Power Measurements

The simplest and most commonly used method for automated measurement of power dissipation in a mobile device uses a precision ammeter. In this traditional method, the device is powered by a low-noise constant voltage source. The precision ammeter, equipped with a serial communication interface, is placed in series with the device's power delivery path. Energy is computed as a function of the measured current and supply voltage. This approach can result in very high accuracy, low bandwidth current measurements, but it is not practical for today's low-voltage devices which typically operate from a single Lithium-Ion cell. During startup, the high in-rush current, I_{in} causes the device's voltage supply, V_{in} to drop due to the relatively large internal ammeter sensing resistance (5 Ω) and its parasitic inductance. In many cases, this drop causes the internal power management protection circuit included in newer devices to suspend startup. Hence, the traditional power measurement technique becomes infeasible, as we experienced first-hand with our transition from an older device to a more modern version.

Instead, we use an improved, non-intrusive power measurement technique, suitable for modern low-voltage devices, which uses a high-bandwidth current-sensor probe clamped around the power supply wire. The current probe is used to measure the battery current based on the magnetic hall-effect. The PDA battery is used instead of the voltage source, since the power management circuitry disables the PDA if the additional battery wires are left unconnected. These wires may be used either for digital communication in advanced "Smart Battery Packs", or simple local analog sensing functions. We connect the current probe to a two-channel oscilloscope through a calibrated amplifier. The battery voltage, V_{in} is also sampled by the oscilloscope using a voltage probe. The sampled current and voltage data is transferred to a PC using a serial interface. The total energy consumed during N samples of the oscilloscope data is calculated using formula (1), where f_s is the oscilloscope sampling frequency.

$$E_{tot} = \frac{1}{f_s} \sum_{i=1}^{N} V_{in}[i] \cdot I_{in}[i] \quad (1)$$

We are currently using a Tektronics TCP312 current probe, a Tektronics TCPA300 amplifier and a Tektronics TDS3032 oscilloscope to record the measurements.

4. EXPERIMENTAL RESULTS

We begin this section by showing preliminary energy measurements for our device in a variety of communication scenarios using GPRS and WiFi. Then, we present wireless communication and energy measurements performed while running our 3 hour traces in a scenario illustrating periodic content refresh. We compare several proxy-based approaches including push and poll-based approaches and the baseline proxyless approach in the two communication scenarios: using WiFi or GPRS alone for all wireless communication. We also investigate the use of SMS messages to signal the device that updates have occured during periods of disconnection. By including important information in the message, such as the number and size of pending updates, we allow the mobile proxy to adaptively use WiFi and GPRS.

4.1 Preliminary Energy Measurements

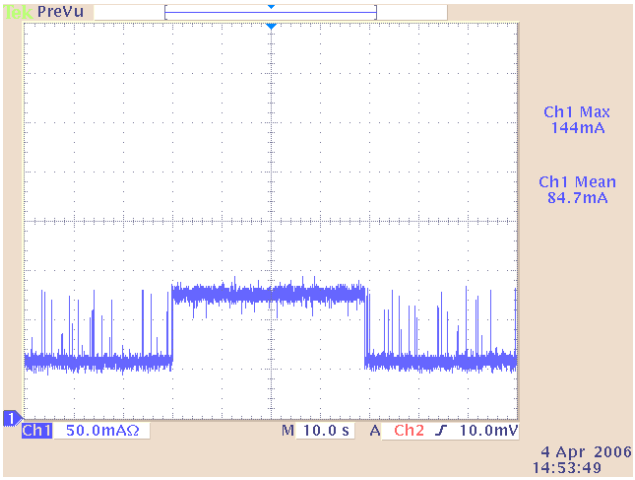
In this section, we characterize the energy consumption of our mobile device hardware. Specifically, we measure the current being drawn by the device as it utilizes each hardware component. In each case, all other non-essential hardware on the device is disabled. For example, when testing the 802.11 connection, we disable the GSM radio and the backlight of the device. We also characterize the cost of downloading data of various sizes over each of the wireless connections.

Figure 3(a) shows the current drawn by the device's processor. The processor transitions from an idle state to an active state, where it continuously processes data for 40 seconds before returning to the idle state. We can see from this screen shot that even while idle, the processor frequently becomes active for short periods of time, probably to handle interrupts and timers. During its active phase, the processor draws an additional 65 mA of current on top of its baseline 65 mA.

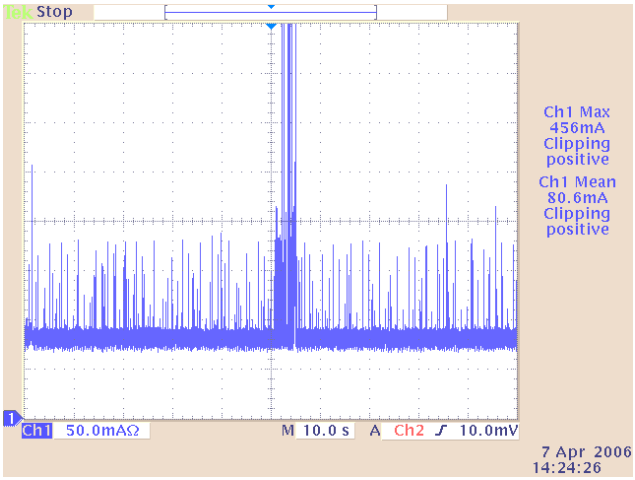
Figure 3(b) shows the current being drawn by the device as it receives an incoming SMS message. The current spikes a couple of times over a 4 second period, with an average increase of 45 mA. We can see from this plot that receiving a SMS messsage is relatively low cost in terms of energy consumption.

Figure 3(c) shows the current drawn by the device as the cellular radio transitions through a number of states. In the beginning, the radio is off, hence the phone connection is disabled. At about the 25 second mark of the experiment, the radio is turned on, resulting in a visible fluctuation in the current drawn by the device as the hardware is powered up and the drivers are loaded. This corresponds to a steady state with an additional current of 20 mA being drawn. At the 50 second mark, we dial the GPRS connection and a data connection is established. There are more fluctuations as hand shaking occurs to establish the connection, before the device settles back to the same level of power consumption as when there was no connection. This experiment shows that little or no energy can be saved by having the cellular radio turned off. Likewise, disabling the GPRS connection results in no energy savings, while there are penalties for connection establishment.

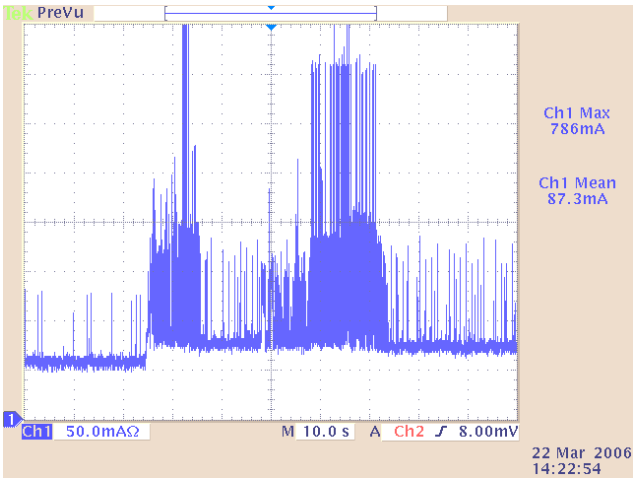
Figure 3(d) illustrates the current being drawn by the device while a 160 KB file is downloaded over the GPRS connection. The current varies considerably over the course of the download, as packets are being sent and received, and as the processor becomes active to process the packets.



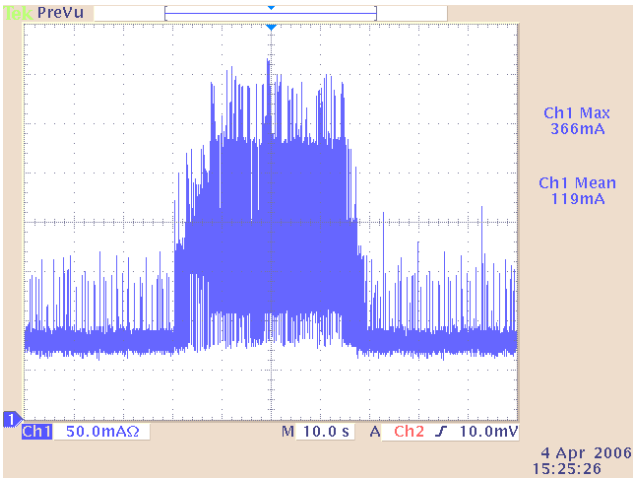
(a) Processor state



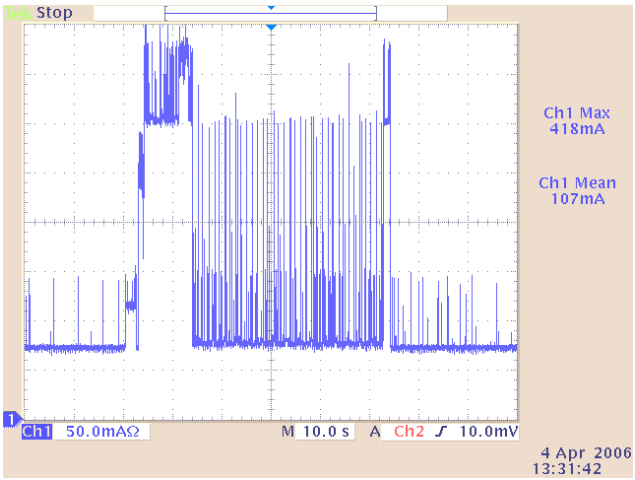
(b) Receiving a SMS message



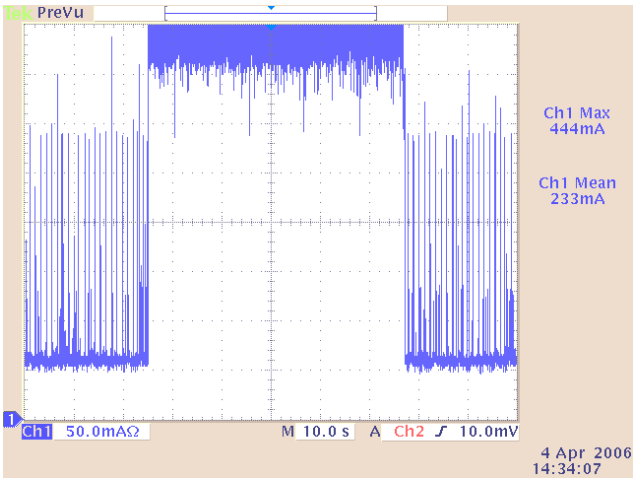
(c) GSM radio state



(d) Downloading a 160 KB file over GPRS



(e) 802.11 radio state



(f) Downloading a 10 MB file over 802.11

Figure 3: Impact of the device state on the current drawn by the device *Scale: 50 mA/div, 10s/div*

On average, over the course of the download, the current increases by 100 mA (an 125% increase over the idle state consumption).

Figure 3(e) plots the current being drawn by the device when the 802.11 radio is switched on and becomes associated with an access point. This causes the current to spike to over 400 mA. After associating with the access point, the network card periodically communicates with the access point to check for incoming data. These periodic checks only require an additional 10 mA on average. The current spikes again just after the 70 second mark as the network card is disabled. This plot illustrates that even though the 802.11 radio is power hungry, aggressively switching it on and off may consume more power in the long run.

Figure 3(f) represents the current being drawn by the device as a 10 MB file is downloaded over the 802.11 wireless connection. During the download, the average current being drawn by the device reaches 375 mA, an increase of over 300 mA (nearly a 450% increase) compared to the idle state.

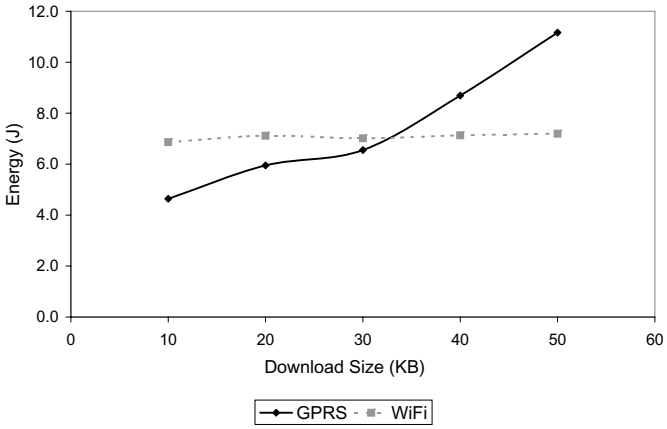


Figure 4: Energy cost of downloading data of various sizes over WiFi and GPRS.

In order to determine the message size for which the energy cost of downloading over GPRS is greater than that of WiFi, we downloaded several file sizes over both connections. In this experiment the device already has an active GPRS connection, since, as shown in Figure 3(c), toggling this connection has little to no energy savings. However we leave the WiFi connection disabled until it is needed. So the cost of downloading over the GPRS connection is just the cost of receiving the file, whereas the cost of downloading it over the WiFi connection also includes the penalty of activating that interface. Figure 4 shows the energy expenditure as a function of file size for WiFi and GPRS. We can see that the cross over point is roughly 30 KB. This experiment tells us that any communication under 30 KB should use the GPRS connection, while for communication over this threshold, the WiFi connection is more energy-efficient.

4.2 Wireless Communication

In order to compare the proxy-based and proxyless configurations in terms of data communication, we ran all configurations introduced in section 3.2, i.e., without a proxy, simple proxy (polling), intelligent proxy (polling), and both a polling and push-based thresholds proxy, on the 3 hour traces described in section 3.1. In each configuration, the mobile client browser requests the URLs encountered in the trace over the 802.11 connection, while the Apache web server returns each saved page from the trace corresponding to the URL. We do not show the results of the experiment for both GPRS and 802.11 because the results are virtually identical. We are measuring the amount of data transferred at the application level, so any minor differences between the control messages transmitted by these two wireless technologies are negligable.

Each of the proxy configurations was set to either poll for updates, or receive pushed updates, periodically with a 4 minute interval. The frequency with which the edge server polls each of the data sources was set to 2 minutes.

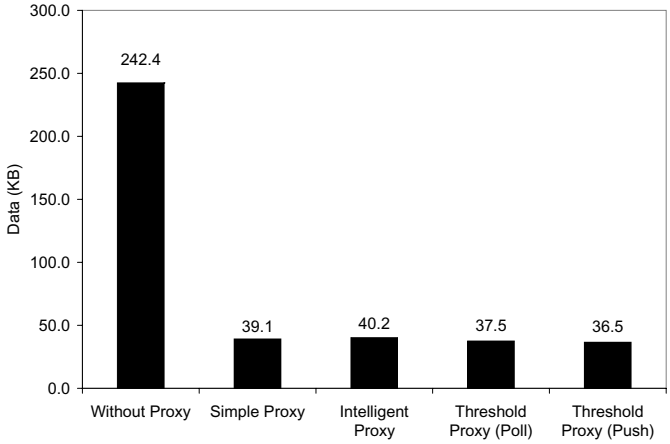


Figure 5: Transmitted Data for the 45 accesses of 4 real web sites

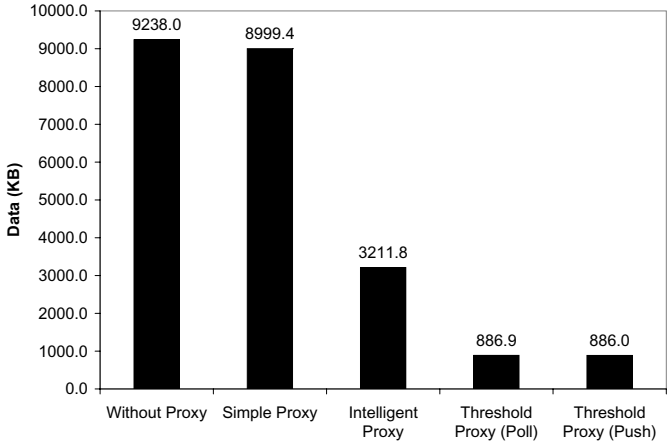


Figure 6: Received Data for the 45 accesses of 4 real web sites

During each three hour experiment, the data is viewed a total of 45 times. The resulting total amount of wireless communication for each configuration is presented in Figures 5 and 6. Table 1 shows the total data transmitted and

	Transmitted (KB)	Received (KB)	Updates	Cache Hits	Misses
Without Proxy	242.4	9238.0	0	0	657
Simple Proxy	39.1	8999.4	220	521	109
Intelligent Proxy	40.2	3211.8	72	512	120
Thresholds Proxy (Poll)	37.5	886.9	11	536	105
Thresholds Proxy (Push)	36.5	886.0	11	536	105

Table 1: Experimental results for each configuration. Cache hits/misses are only for mobile proxy and not the web browser's cache. Updates are the number of page changes that occurred, several updates may be sent in one batched transfer.

received in conjunction with the total number of updates and cache hit statistics of the mobile proxy cache.

Figure 5 illustrates the amount of wireless data that was transmitted from the PDA over the 45 accesses. We can clearly see that the amount of data that the PDA is required to send is greatly reduced in all proxy configurations compared to the proxyless configuration. The reason is that, for the proxyless configuration, each time the client wants to view the data, each of the HTTP requests must be sent in their entirety over the wireless link. In contrast, in any poll-based proxy configuration, the mobile proxy needs to send, at most, two 28 byte packets to the edge server proxy: an update inquiry packet and an update acknowledgement packet (only if an update has occurred). This is because, after the first round of communication, the edge server proxy knows exactly what the client is interested in, and as a result the client no longer needs to send the entire request. In the push-based proxy configuration, the mobile device saves an additional 28 byte packet per request, since no update inquiry message needs to be sent by the mobile device. This contributes to the small reduction of 2.7% for the data transmitted in the push-based approach compared to its poll-based counterpart.

In contrast to the data transmission graph, Figure 6 illustrates the significant differences between the various proxy configurations in terms of data received over the wireless link during the 45 accesses of our trace. We can see that the simple proxy saves only 2.6% of the data over the proxyless configuration. This is because, as discussed in section 3.1, each of the sites in the trace changes rather frequently. In particular, the content on all sites changes at least once during each 4 minute polling interval of the experiment. As a result, the simple proxy method downloads nearly the entire batch of data each period.

The intelligent proxy reduces much of the wireless data received by reducing the number of updates the edge server proxy sends to the mobile client. As we can see from Table 1, by only sending updates when parts of the page of interest to the user change, we reduce the total number of updates by a factor of 3. This translates into a 65.2% reduction in the amount of data received when compared to the baseline proxyless approach.

Finally, the thresholds proxy reduces the amount of data received over the wireless link even further. By setting reasonable thresholds for numerical changes that warrant sending an update to the client, we reduce the number of updates by a factor of 20 when compared to that of the simple proxy. For example, by setting a threshold for the change, we prevent the edge server proxy from sending several dozen updates while the value of the Euro fluctuates between $1.22736 and $1.22740. This drastic reduction in the number of updates sent results in an order of magnitude savings in the amount of data received over the wireless link. Again the push based approach offers marginal savings in comparison to its poll-based counterpart. In particular, the push-based proxy registers a negligible 0.1% savings as a result of not having to receive the 28 byte "No Updates Available" packet that the polling approach incurs in the situations where no content change has occurred.

4.3 Energy Consumption

Figure 7 presents the average energy consumed by the device per download period (i.e., loading each of the 4 web sites in the browser) for the 3 hour experiments presented in the previous section, in six configurations: the baseline proxyless configuration using the 802.11 connection and using the GPRS connection, our poll-based thresholds proxy configuration using the 802.11 connection and using the GPRS connection, and our push-based thresholds proxy configuration using the 802.11 connection and using the GPRS connection.

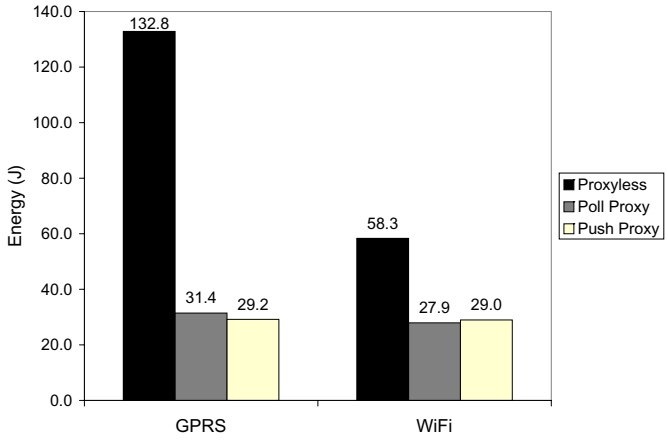


Figure 7: Average energy expenditure per download period

We can see that all configurations of the thresholds proxy are superior in energy conservation compared to their proxyless counterparts. Our proxy system reduces energy costs by factors of 2.1 and 4.5 when used over the 802.11 and GPRS connection, respectively.

4.3.1 Energy Consumption in Push versus Poll Proxy

As illustrated in Figure 7, the differences in energy consumption between the push-based and poll-based proxies are small for both WiFi and GPRS. The push-based proxy using the GPRS connection conserves 7% energy per download period compared to the poll-based proxy. Maintaining a persistent connection with the edge server in the push-based configuration is more energy-efficient than requiring the device to create a connection, request an update, and tear down the connection during each period in this case. The push based proxy using the WiFi connection on the other hand, uses 4% more energy per download period than it's polling based counterpart. Constantly listening for incoming communication over the WiFi connection requires more energy than periodically sending update request packets. This push based proxy could potentially save more than the polling approach, if the device used a small listening window for receiving updates. However, the need to synchronize the clocks of both proxies over the WWAN makes this approach unattractive.

4.3.2 Energy Consumption for Off-line Updates Using the Hybrid Approach

In this section, we analyze the energy consumption of the SMS based proxy system described in Section 2.3. The mobile proxy requests an update only when it receives an SMS message specifically informing it that there are updates available. The proxy uses the control information contained in this SMS message to determine the best interface to use for downloading the update. The proxy uses GPRS to download any updates under 30 KB and WiFi for updates over this threshold. This value was based on the results shown in Figure 4.

The average energy consumption of this proxy is illustrated in Figure 8 along with the best results for the GPRS and WiFi only proxies. The hybrid SMS proxy saves an additional 14% energy over the push based GPRS proxy and 10% over the polling WiFi proxy. The savings are a result of not having to send periodic update requests, or conversely, listening over the wireless channel for incoming updates, and from using the most efficient download method for acquiring the updates when they are available.

4.3.3 Energy Consumption for New Page Accesses

To determine the energy consumption for the case of visiting new pages i.e., cold cache miss, we ran an experiment where we viewed one of the pages in our trace with an empty browser cache and an empty proxy cache. The page used in this experiment contained 51 embedded files, consisting of dozens of small images, a couple of style sheets, and several javascript files. The HTML page and all of its embedded files were 185 KB. We used a proxyless setup as baseline for comparison.

The total energy required to view the selected page in each configuration is illustrated in Figure 9. Compared to the proxyless approach, the energy expenditure is reduced by 69% when using the GPRS connection in conjunction with our proxy system, and by 15% when using the WiFi connection. These savings are a result of the prefetching and batching that the edge proxy performs for all embedded objects in the HTML page.

Figures 10 and 11 illustrate the energy savings for each communication method during the download period. Since

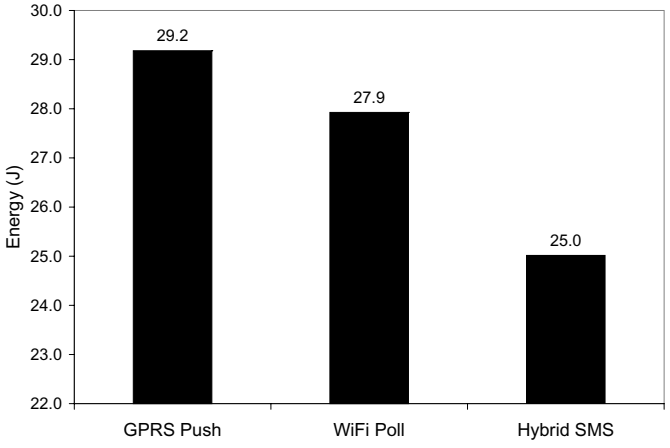


Figure 8: Average energy expenditure per per download period

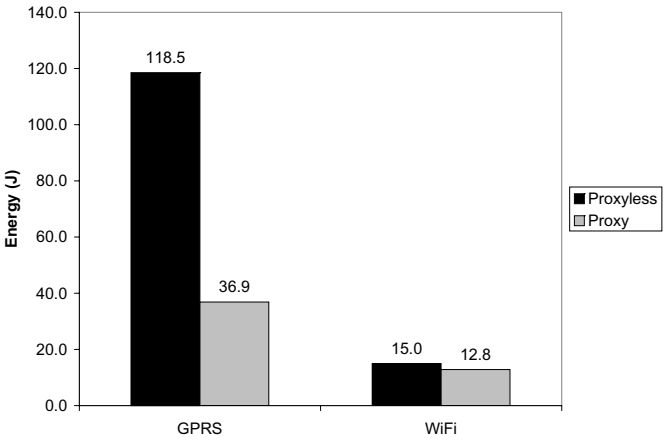


Figure 9: Total energy costs of downloading a cold page

the WiFi connection is high bandwidth and low latency, the potential savings from prefetching and batching are minimal. However, we save 15% of the consumed energy by limiting the number of times the power hungry WiFi card needs to be used, as illustrated in Figure 10. Even though it takes our proxy system several seconds longer to load the page, after one round of communication, our mobile proxy is able to serve the remainder of the browser's requests locally. This results in energy savings overall. In contrast, the proxyless approach makes 52 individual requests over the WiFi connection.

The prefetching and batching behavior of the proxy system is much more beneficial when browsing over GPRS. As illustrated in Figure 11, the energy savings come from being able to load the page in nearly a third of the time when using the proxy. By downloading the web page and all its embedded files in one transfer, we are bypassing the large round trip time of the GPRS connection.

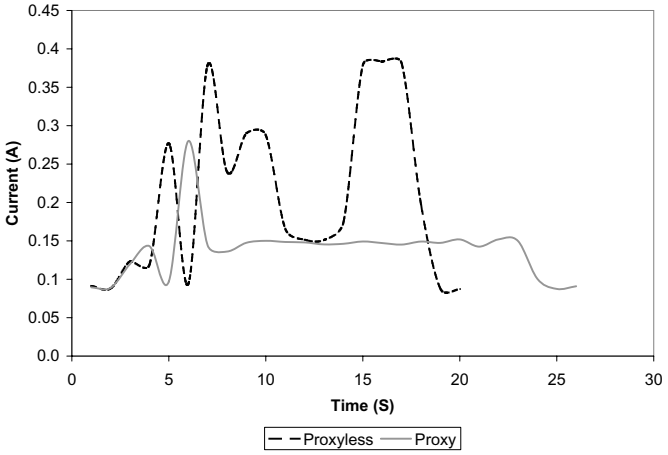


Figure 10: Downloading a new page over WiFi

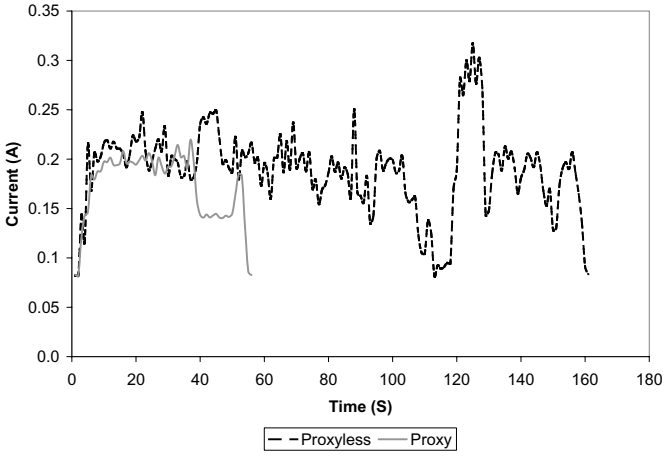


Figure 11: Downloading a new page over GPRS

5. DISCUSSION

Our results have shown that our proxy-based approach achieves considerable savings in terms of both data communicated over the wireless link and battery consumption on the mobile device for both 802.11 and GPRS connections. In the GPRS scenario, the proxy is able to bypass much of the latency associated with the slow cellular link. On the other hand, the same proxy mechanism, when used over an 802.11 connection reduces the amount of time that the high energy network interface must be in use.

The push-based approaches have negligible data transfer savings, as well as negligible energy savings, compared to their poll-based counterparts. Moreover, we argue that push-based approaches using WiFi are challenging to implement in practice. First, a push-based approach assumes always-on semantics for clients, which is currently supported on cellular phones and Blackberries [7], but not over WiFi. In practice, due to power considerations, push-based communication is implemented by following a very tightly synchronized schedule for client and server. Otherwise, the client wastes power while waiting in idle mode for the server to contact it. For a push-based notification system to support WiFi devices, having an agreed upon communication schedule is not enough. The system has to also include a facility to buffer messages, as a mobile WiFi device may go offline at any time, as the user moves between hotspots. Second and more importantly, a push-based approach requires mobile IP [8] support; the server has to have a way of contacting the client, while the mobile client's IP address is potentially changing as it connects from different access points.

The SMS based proxy is the most promising configuration. Most carriers offer free incoming SMS messages. Furthermore, receiving an SMS message has minimal energy consumption (as shown in Figure 3(b)). As we have shown, this method conserves the most energy overall. Furthermore, it has the added benefit of being able to deliver content change notifications to the user, even if the user is in a disconnected state.

Finally, there is an obvious trade-off between the data freshness and the cost of maintaining it. Frequently pulling updates from the edge proxy implies high battery consumption on the mobile device. In our approach, the appropriate compromise is left up to the user. Our approach maximizes the battery savings for a particular data freshness time threshold set by the user. While we do not currently offer anything beyond the basic browser-based interface described in section 2.1, we can envision an adaptive approach where the update pull interval may be decreased or increased by the user at the press of a button. For instance, if the user anticipates an imminent period of disconnection, they may temporarily increase the update interval for their content.

6. RELATED WORK

Several other works, most notably WebExpress [9] and PAWP [10], have investigated using proxies to reduce energy consumption and/or perceived latency when browsing the web on a mobile device.

WebExpress uses two intercepts (proxies), one located on the mobile client and one on the server side. These two proxies use many of the same, or similar, techniques we use to reduce the costs of browsing the web; caching, protocol reduction (using one wireless connection for requests as opposed to creating multiple), and HTTP header reduction. WebExpress does not prefetch and batch embedded objects when new requests are received, and cache updates must be made on an individual item by item basis. Most importantly WebExpress is not as user centric as our system, it does not keep client profiles, and does not update the cache based on the client's interests. As we have shown in Section 4.2, our user centric approach has a dramatic reduction on the amount of wireless communication that occurs.

The authors of PAWP [10] use a single proxy near the wireless access point to buffer WWAN web traffic to/from the device so that they can schedule the data to be delivered to the mobile client in alternating bursts of high and no activity. This allows the WiFi card on the device to have a more aggressive sleep pattern, which in turn conserves energy. Unlike our solution, the client is still required to make each individual request and download each individual file. The PAWP proxy does not cache any data, it only gathers it on request. Also, by not filtering out redundant data, the PAWP proxy results in a much larger amount of wireless communication than our proxy solution.

The authors of [11] provide a comparative performance study of various techniques, at the application, session, transport and link layers, to improve web browsing latencies over GPRS. They examine both proxy based and proxy free solutions and conclude that proxy based approaches offer the greatest improvements. This is mainly due to the proxy's ability to use techniques such as extended caching, delta encoding, prefetching and batching.

Sinclair et al. [1] provide a method for conserving power by separating the data and control signals and using a wake event on the control line to wake up the device upon an incoming connection. This mechanism requires specialized hardware. In contrast, in our situation we use commodity devices with no modifications and exploit client profile information in order to help conserve power.

Our work builds on the concepts of user profiles and data recharging [12, 13] that have been recently proposed to enable disconnected operation of mobile clients. However, existing research in this area focuses on user profile language specifications [12, 13] and on algorithms to be run by the service provider to optimize the creation and maintenance of super-profiles that combine the preferences of many users [13]. Similarly, most underlying algorithms in publish/subscribe distributed systems [14] investigate efficient filtering when scaling a system of publishers and distributed brokers to supporting millions of subscriptions and of filtering hundreds of new events per second. Such systems are typically using evaluations based on simulation to investigate scaling with the number of clients.

While our work shares similarities with these systems, we adopt a user-centric approach and evaluation methodology. We avoid introducing new languages or complex interfaces that may prevent wide acceptance. Instead, we concentrate on providing the maximum benefits to the user in terms of cost, battery life and convenience through a profile-driven infrastructure with seamless integration into already familiar applications and environments.

Push-based solutions for specialized types of information such as Bell's e-mail notifications [15], Research in Motion's Blackberry [7], location-based services for taxi availability [16] or food/entertainment advertisements [17] already exist or are being proposed. However, most types of information that are being currently pushed are simple text messages such as e-mail notifications, instead of the content the user is actively browsing.

A scheme for cutting down on network traffic by pushing content to the users is also presented by Baker et al. [18]. The main difference in their approach, however, is that the content itself is not pushed transparently to the user, only a message containing the URL of the new content is. It is up to the user to determine whether or not to fetch that content.

A variant of a push-based scheme for data refresh similar to our own appears in related work on broadcast disks [19, 20]. Broadcast disks are indexing algorithms for data that is being broadcasted over a wireless channel. Mobile nodes tune in to listen to the index being broadcasted. The index, which is assumed to be much smaller than the full data broadcast, identifies the time when a given data item will be transmitted. A node then knows when to wake up to catch the transmission of an item of interest. While these systems have similar goals to our own, they do not target data refresh for dynamic content browsing. Furthermore, in their case, the server maintains no knowledge of the content the user already has.

RSS [21, 22] is a system with which web servers can "push" content directly to the users over the internet. This system involves the user subscribing to that particular web server's content and then each change appears on the user's desktop. However, the underlying principle of this system is that the client regularly polls the site automatically. This gives the illusion of a push based system, but underneath the system is an automated web polling service.

Our centralization of update processing on the proxy is similar in spirit to the cyber-foraging approach [23] of offloading computation and storage from mobile devices to available resource-rich nodes on the Internet in order to save power and money.

Many schemes have investigated communication savings for web content serving such as through server directed transcoding [24], and optimistic deltas [25]. Our scheme is orthogonal to these techniques and could be used in conjunction with them for further data savings. Our approach determines when to push modifications, while the optimistic delta approach and the transcoding approaches focus on sending smaller updates (e.g., optimistic deltas sends just the modified portions of the page instead of the whole page).

7. CONCLUSIONS

We introduce a novel approach to transparent, automatic data refresh for mobile devices. Our approach is centered around a general purpose mechanism for letting the user specify her interest in changes to specific parts of pages. We avoid introducing new languages or complex interfaces that may prevent wide acceptance. Instead, the user loads her favorite pages on her mobile device browser and highlights areas of interest in those pages using the regular browser's cursor. We offload the detection of updates to content that matches the user's interest, onto a fully-connected edge proxy. Subsequently, either while the client is actively browsing or while attending to everyday activities of travel, shopping, work and play, the mobile device performs automatic data refresh transparently to the user.

Our approach is fully implemented using both WiFi and GPRS communication on an actual mobile device and evaluated on real world data traces. Our results show that our general purpose proxy system saves data transfers to and from the mobile device by an order of magnitude and battery consumption by up to a factor of 4.5. These savings are due to the fact that, typically, there are frequent changes to parts of dynamic content web pages that the user is not interested in, such as the time of day or an ad banner. In addition, many changes in the n-th decimal of numerical values can be typically ignored. We have shown that, a push-based approach provides minimal gains over a poll-based approach. Additionally, we have shown that by using the existing SMS infrastructure to deliver notifications on dynamic content changes, we can offer an energy efficient and user friendly way to keep the clients up to date with their content of interest.

8. REFERENCES

[1] E. Shih, P. Bahl, , and M. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *ACM/IEEE International*

Conference on Mobile Computing and Networking (Mobicom), 2002.

[2] "Ebay.ca," http://www.ebay.ca/.

[3] "Cnn.com weather, toronto, on," http://weather.cnn.com/weather/forecast.jsp?locCode=YYZ.

[4] "Xe.com," http://www.xe.com/.

[5] "Yahoo! finance," http://finance.yahoo.com/.

[6] Thomas A. Phelps and Robert Wilensky, "Robust intra-document locations," in *Proceedings of the 9th international World Wide Web conference on Computer networks*, Amsterdam, The Netherlands, The Netherlands, 2000, pp. 105–118, North-Holland Publishing Co.

[7] Research in Motion, "Blackberry," http://www.blackberry.com.

[8] C Perkins, "IP Mobility Support," RFC 2002, Oct. 1996, ftp://ftp.isi.edu/in-notes/rfc2002.txt.

[9] Barron C. Housel and David B. Lindquist, "Webexpress: a system for optimizing web browsing in a wireless environment," in *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, 1996, pp. 108–116.

[10] Marcel C. Rosu, C. Michael Olsen, Chandrasekhar Narayanaswami, and Lu Luo, "Pawp: A power aware web proxy for wireless lan clients.," in *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.

[11] Rajiv Chakravorty, Suman Banerjee, Pablo Rodriguez, Julian Chesterfield, and Ian Pratt, "Performance optimizations for wireless wide-area networks: comparative study and experimental evaluation," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, New York, NY, USA, 2004, pp. 159–173, ACM Press.

[12] R. Agrawal and E. L. Wimmers, "A framework for expressing and combining preferences," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, August 2000.

[13] Mitch Cherniack, Eduardo F. Galvez, Michael J. Franklin, and Stan Zdonik, "Profile-driven cache management," in *International Conference on Data Engineering (ICDE)*, 2003.

[14] Francoise Fabret, H.Arno Jacobsen, Francois Llirbat, Joao Pereira, Kenneth Ross, and Dennis Shasha, "Filtering algorithms and implementation for very fast publish/subscribe systems," in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, 2001, pp. 115–126.

[15] Bell Canada, "Blackberry 7750 wireless handheld," http://www.bell.ca/shop/en_CA_BC/Sme.Sol.Wireless.Solutions.BlackBerry.page.

[16] Zingo Taxi, "Location based services," http://www.springwise.com/newbusinessideas/2003/09/zingo_taxi.html.

[17] I. Burcea and H.A. Jacobsen, "L-topss - push-oriented location-based services," in *4th VLDB Workshop on Technologies for E-Services (TES'03)*, 2003.

[18] Shaun Baker, "Active internet services: Pushing content to the people," in *whitepaper for SIP development group of Siemens Switzerland*, 2000.

[19] Swarup Acharya, Rafael Alonso, Michael Franklin, and Stanley Zdonik, "Broadcast disks: data management for asymmetric communication environments," 1995, pp. 199–210.

[20] A.J. Xu, W. Lee, and X. Tang, "Exponential index: A parameterized distributed indexing scheme for data on air," 2004.

[21] "Xml.com: Rss description," http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html.

[22] Mel Blackman, "Pushing web content with really simple syndication," *e-Pro Magazine*, 2001.

[23] R. Balan, J. Flinn, M. Satyanarayanan, S. Sin, and H. Yang, "The case for cyber foraging," 2002.

[24] Bjorn Knutsson, Honghui Lu, Jeffrey Mogul, and Bryan Hopkins, "Architecture and performance of server-directed transcoding," *ACM Transactions on Internet Technology*, vol. 3, no. 4, pp. 392 – 424, November 2003.

[25] Gaurav Banga, Fred Douglis, and Michael Rabinovich, "Optimistic Deltas for WWW Latency Reduction," in *Proceedings of the 1997 USENIX Technical Conference*, 1997.