Dynamic Data Updates for Mobile Devices by Using 802.11 Wireless Communications

B. V. Ramanamurthy[1], K. Srinivas Babu[2] and **Mohammed Sharfuddin[3]**

Submitted in May 2010; Accepted in December 2010

Abstract - Mobile devices are used for conveying important information. Opportunity exist to introduce users to different application of resource constraint mobile device 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 his 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 change 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.

Index Terms - Mobile wireless communication, proxy Process, caching, pre fetching, energy measurement.

1.0 INTRODUCTION

In these 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 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 each page that are of interest to them, ii) off loading 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 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. 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 Wi-Fi (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. 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.[1][2].

2.0 OUR FRAMEWORK

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

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.

[1,2,3]*Department of Compute Science and Engineering, Guru Nanak Engineering College, Ibrahimpatnam, R. R Dist, Andhra Pradesh, INDIA*
E-Mail: [1]drbvrm@gmail.com and [3]sharfuddin_se@yahoo.com

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 . 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.

2.2 Architecture Components

There are 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. 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.

2.2.1 Mobile Device Components

1) Mobile device web browser
2) Mobile proxy
3) Cache
4) Hardware Manager

2.2.2 Edge Server Components

1) Edge server proxy
2) Cache Manager
3) Update Manager
4) Cache

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 servers 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] [7].

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 pre fetches 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[3][6].

- 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. [4]

3.1 Proxy-Based 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.

3.1.1 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.

3.1.2 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.[7]

3.1.3 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.

3.1.4 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. [2][5][7]

3.2 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 Wi-Fi or GPRS.[5][6]

3.3 Experimental Setup for Power Measurements

The 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, Iin causes the device's voltage supply, Vin to drop due to the relatively large internal ammeter sensing resistance (5□) 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.

Calculated using formula where fs are the oscilloscope sampling frequency [2][8]

$$Etot = 1/fs * \sum_{i=1}^{n} Vin[i].Iin$$

	Transmitted (KB)	Received	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	530	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. In contrast to the data transmission graph, 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, [2][7]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 proxy less approach. Finally, the thresholds proxy reduces the amount of data received over the wireless link.

3.3.1 Energy Consumption

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, the baseline proxy less 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.

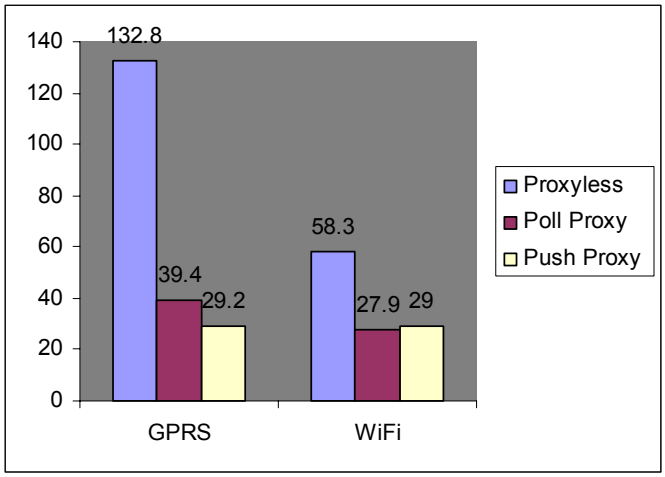

Figure 1: Average Energy Expenditure per Download Period

We can see that all configurations of the thresholds proxy are superior in energy conservation compared to their proxy less 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.

3.3.2 Energy Consumption in Push Versus Poll Proxy

As stated in Fig 1, [2] the differences in energy consumption between the push-based and poll-based proxies are small for both Wi-Fi 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 Wi-Fi 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 Wi-Fi 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 [9][10].

4.0 ENERGY CONSUMPTION FOR OFF-LINE UPDATES USING THE HYBRID APPROACH

In this section, we analyze the energy consumption of the SMS based proxy system. 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 Wi-Fi for updates over this threshold.

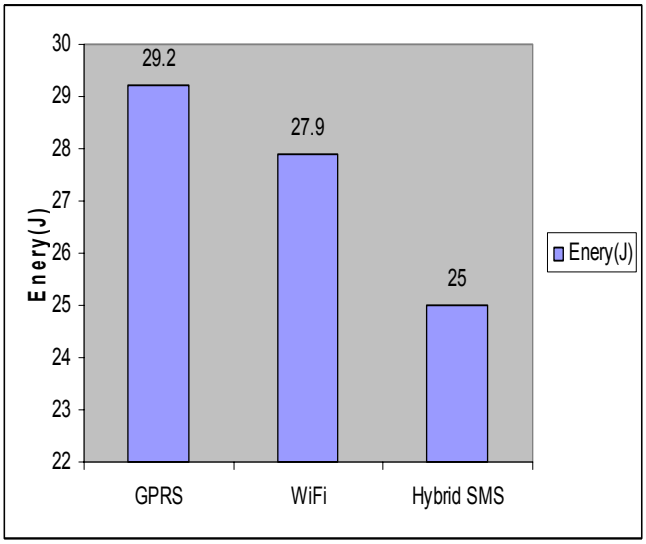

Figure 2: Average Energy Expenditure per Download Period

The average energy consumption of this proxy is stated in Fig 2, [2] along with the best results for the GPRS and Wi-Fi only proxies. The hybrid SMS proxy saves an additional 14% energy over the push based GPRS proxy and 10% over the polling Wi-Fi proxy. The savings are the 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.[7]

4.1 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 age used in this experiment contained 51 embedded files, consisting of dozens of small images, a couple of style sheets, and several JavaScript files. We used a proxy less setup as baseline for comparison. The total energy required to view the selected page in each configuration is Compared to the proxy less approach, the energy expenditure is reduced by 69% when using the GPRS connection in conjunction with our proxy system, gives 15% when using the Wi-Fi connection see fig 3,[2] and 12.8 for proxy system see fig 3[2].

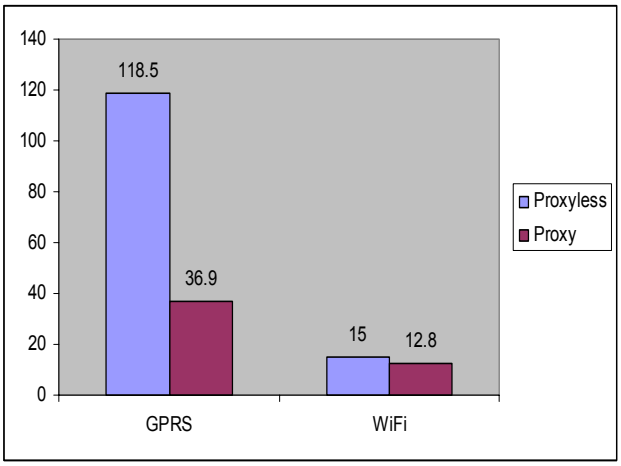

Figure 3: Total Energy Cost for Downloading a Page

5.0 CONCLUSIONS

In these we introduced an automated approach to 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 Wi-Fi 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.

REFERENCES

[1]. E. Shih, P. Bahl, and M. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in IEEE International Conference on Mobile Computing and Networking (Mobicom), 2008.

[2]. Armstrong, T., Trescases, O., Amza, C., de Lara, E. Efficient and transparent dynamic content updates for mobile clients.Proceedings of the 4th international conference on Mobilesystems, applications and service, pages 56-68. 20066]

[3]. Thomas A. Phelps and Robert Wilensky, "Robustintra document locations," in Proceedings of the 9th international World Wide Web conference on Computer networks, Amsterdam, The Netherlands, The Netherlands, 2007, pp. 105-118, North-Holland Publishing Co.

[4]. ResearchinMotion,"Blackberry," http://www.blackberry.com.

[5]. C Perkins, "IP Mobility Support," RFC 2002, Oct. 2005, ftp://ftp.isi.edu/in-notes/rfc2002.txt.

[6]. Barron C. Housel and David B. Lindquist, "Web express: 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, 2000, pp. 108-116.

[7]. 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.

[8]. 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.

[9]. 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.

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