Reducing the Delay and Power Consumption of Web Browsing on Smartphones in 3G networks

Bo Zhao, Byung Chul Tak and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
Email:{bzhao, tak, gcao}@cse.psu.edu

***Abstract*—Smartphone is becoming a key element in providing greater user access to the mobile Internet. Many complex applications, which are used to be only on PCs, have been developed and run on smartphones. These applications extend the functionalities of smartphones and make them more convenient for users to be connected. However, they also greatly increase the power consumption of smartphones and many users are frustrated with the long delay of web browsing when using smartphones. In this paper, we have discovered that the key reason of the long delay and high power consumption in web browsing is not due to the bandwidth limitation most of time in 3G networks. The local computation limitation at the smartphone is the real bottleneck for opening most webpages. To address this issue, we propose an architecture, called Virtual-Machine based Proxy (VMP), to shift the computing from smartphones to the VMP. To illustrate the feasibility of deploying the proposed VMP system in 3G networks, we have built a prototype using Xen virtual machines and Android Phones with T-Mobile UMTS network. Experimental results show that compared to normal smartphone browser, our VMP approach reduces the delay by more than 80% and reduces the power consumption during web browsing by more than 45%.**

I. Introduction

Due to the arrival of a new generation of improved smartphones, with fuller sized screens, improved user interfaces, and advanced features such as GPS and accelerometers, smartphone is becoming a key element in providing greater user access to the mobile Internet. Many complex applications, which are used to be only on PCs, have been developed and run on smartphones. These applications extend the functionalities of smartphones and make them more convenient for users to be connected. However, they also greatly increase the workload on smartphones and drain the smartphone battery. In addition, many users are frustrated with the long delay of web browsing when using smartphones.

There have been some research on optimizing the battery power [1]–[3]. However, these research work is limited to reducing the power consumption of one component of the smartphone such as display, wireless interface [3], or some network protocol. From the web server's point of view, smartphone is treated the same as other computing devices such as desktops at office. However, there is a fundamental difference between smartphones and desktop PCs such as display size and battery power. To address the fundamental difference between smartphone and desktop PC, some web servers adopt transcoding techniques [4], [5], where smartphones are treated differently. For example, if the request is from a smartphone, the image resolution is reduced to fit the smartphone, and some javascripts are disabled to conserve the battery power of the smartphone. However, to use such techniques, the web server has to be re-designed, and hence only a limited number of web servers can use these techniques. In addition, some techniques [6], [7] applied the optimization approaches on the proxy. Flash proxy [6] translates flash content to other components on a proxy for transparently supporting it on mobile browsers that do not have built-in support. Opera Mini [7] is a commercial product which uses multiple transcoding approaches to reduce the traffic load of the browsing service on smartphones to save money on data charges.

Even with transcoding techniques, the delay and power consumption problems cannot be fully addressed. This is because 3G network has much more bandwidth compared to 2G or 2.5G network, and bandwidth is not really the bottleneck for most webpages although it is an important factor. We found that the smartphone CPU workload can easily reach 100% for a long period of time during web browsing and hence the local computation power is the real bottleneck for most webpages. As a result, we need to design an architecture for smartphones to access the mobile Internet.

In this paper, as shown in Figure 1, we propose an architecture to shift away the computation extensive web browsing from smartphones to reduce the delay and power consumption. In this architecture, a proxy is added between the web server and the smartphones, and a new client is added on the smartphone side to interact with the proxy. For the smartphone, instead of sending request to the web server, the request is sent to the proxy through the wireless network. The proxy sends the request to the web server and gets reply from it. Then, the proxy sends the reply to the smartphone through the wireless network. At this time, the proxy can run different optimization techniques to reduce the resource consumption at the smartphone. For example, the proxy can act like a real web browser which displays the content of the webpage. Instead of sending the original HTTP reply which contains many CPU intensive objects, it simply does a screen copy which copies the screen display of the browser to the smartphone. The proxy can also do some other adaptation based on the battery power level and the wireless signal level of the smartphone. The proxy can be anywhere in the Internet, but can achieve better performance if it is inside the telecommunication network. Due to security

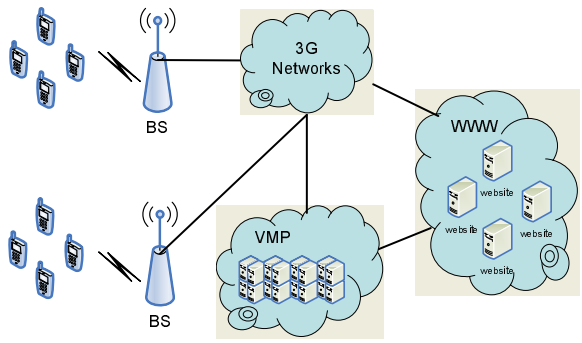

Figure 1. The VMP architecture. A web request from the smartphone will be sent to the proxy which forwards the request to the web server and gets reply from it. Then, the proxy does a screen copy and sends the data to the smartphone through the wireless network.

and privacy issues, the proxy starts a virtual machine (VM) for each smartphone request and releases the resource after the request is served.

Different from existing work on proxy and transcoding [4]–[6], which are limited to dealing with images and multimedia content, our proxy based architecture shifts computing from smartphones to the VMs on the proxy. Thus, we refer to our architecture as VM-based Proxy (VMP). Through the VMP architecture, dynamic content (javascript, flash, etc) can be run at the server side, and hence saving the power of smartphones.

To illustrate the feasibility of deploying the proposed VMP system in 3G networks, we have built a prototype using Xen virtual machines and Android Phones with T-Mobile UMTS network. The experimental results show that compared to normal smartphone browser, our VMP approach reduces the delay by more than 80% and reduces the power consumption during web browsing by more than 45% on average. For browsing content rich commercial webpages, our VMP approach can reduce the delay by more than 92%, and reduce the power consumption of web browsing by more than 58%.

The paper is organized as follows: Section II discusses related work. Section III presents the motivation of our work. Section IV presents the design of our virtual machine based proxy. In Section V, we present our testbed and experimental results. Section VI discusses the limitation of our work. Finally, Section VII concludes the paper.

II. Related Work

Recently, research on reducing the power consumption of smartphones has received much attention. Most of them focus on optimizing the power usage of individual components of smartphones, such as user interface [1], applications [2], etc.

To address the computing limitation of smartphones, remote execution approaches [8] [9] [10] have been developed. Chun and Maniatis [8] proposed a new architecture to augment capabilities of smartphones by moving execution of resource expensive applications to cloud computing. Cloudlets [9] uses nearby remote server and thin-client technique to virtually extend the computing capability of a smartphone, so it can support computing intensive applications. MAUI [10] enables fine-grained energy-aware offloading of mobile code to the infrastructure to save power. These remote execution approaches [8]–[11] mainly focus on future computation intensive applications (e.g., natural language processing, face recognition, etc) which are rarely used currently. However, our work focus on shifting computation expensive web browsing applications such as flash or javascript to remote infrastructure to save power. Since web browsing is widely used by many users, most users can benefit from our work.

Thin-client [12] [13] provides an alternative remote execution approach for web browsing service on PDA. Thin-client computing was originally designed to offload local computing by increasing network traffic to eliminate the user delay experience caused by the remote execution. Therefore, thin-client computing is not designed for power efficiency, and existing work [13] [9] only focuses on reducing the delay with WiFi which has short latency and high bandwidth. Although both our work and thin-client computing have the screen copy function, our architecture addresses the power saving issue in 3G which has long latency and slow data transfers, and provides extensive dynamical adaptation, security protection and cloud based service management.

In many research projects and commercial systems, proxies are used to adapt content to better fit the network, hardware and software characteristics of the mobile devices [5], [6]. Transcoding techniques are used to transform an image object [14] or video object [15] from one version to another. However, existing work on transcoding and proxy is limited to deal with image and multimedia content. At that time (about 5 to 10 years ago), dynamic web content such as flash and javascripts are not very popular, which are the focus of our solution. Through our VMP architecture, the performance of web browsing is improved by shifting the dynamic web content such as flash and javascript to run on the proxy, and getting the display data to the smartphones. In addition, Opera Mini [7] applied other transcoding approaches to reduce the traffic load for saving money on data charges, instead of the power consumption. For example, its mechanisms on the smartphone side need communicate with proxy frequently when user reads the pages, so it consumes a lot of power. However, Our VMP focuses on saving power consumption and delay, but not the traffic load.

III. Motivation

In 2G or 2.5G networks, due to the low network bandwidth, data transmission delay is the major reason for the long delay in web browsing. However, 3G network has much larger bandwidth (at least 100KB/s in our case), and then the data transmission should not be the major reason of the long delay in web browsing although this is contradictory

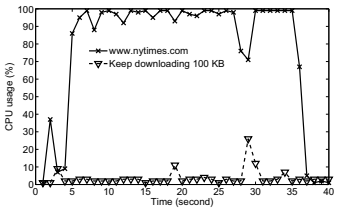

Figure 2. The CPU usage of opening a webpage or repeatedly downloading 100KB of data

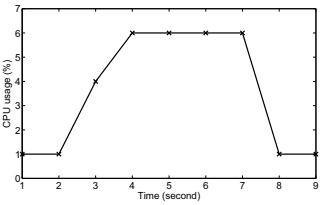

Figure 3. The CPU usage for downloading 522 KB of data

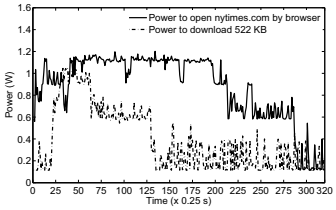

Figure 4. The power consumption of web browsing and data downloading

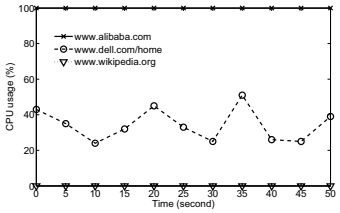

Figure 5. The CPU usage after being opened the webpages on smartphone

to our common knowledge. From various measurements we found that the CPU workload can easily reach 100% for a long period of time during web browsing, and hence the local computing capability on modern smartphones (Android, iPhone, etc) is the real bottleneck for opening most webpages.

Part of the reason is because the webpages become more complex and contain more objects each year. Since the web appeared in 1995, the number of objects per page has grown by 21.7 times [16]. From 2003 to 2008, the number of objects in an average webpage has been nearly doubled from 25.7 to 49.9. Thus, it requires a lot of CPU power to open modern webpages. Moreover, the dynamic web content such as javascripts and flash, is widely used in many webpages. Although desktop PCs can easily handle these dynamic web content, it takes significantly amount of time and power for smartphones to run these javascripts and flash, and open these content rich webpages.

We measured the CPU usage of opening webpages on an Android smartphone (detailed experimental setup will be shown in Section V.) Figure 2 compares the CPU usage of two cases: One is to open "www.nytimes.com", and the other is to keep downloading 100KB of data. To download data, we develop a small program, which sets up a socket connection between the smartphone and the server. The smartphone sends one byte of data to the server, which sends back 100KB of data. After receiving this 100KB, the smartphone repeats the process again. As shown in the figure, to open a webpage, the smartphone CPU workload reaches almost 100% most of the time. After the webpage is opened (at time 37), the CPU workload is reduced to about 4%. This figure also shows that downloading data does not use too much CPU. Thus, we believe the local computation related to web browsing is the bottleneck, which is the key reason for the long delay in web browsing.

Figure 3 further verifies this result. Since opening www.nytimes.com only needs to download 522 KB of data, we investigate how much CPU the smartphone needs to download 522 KB of data. As shown in the figure, it only costs 6% CPU, and it takes about 5.4 seconds to download 522 KB of data, but it takes more than 30 seconds for the web browser to open the webpage with the same amount of data.

To further verify that local computation is the bottleneck, we also used a laptop and Android smartphone to connect to the same WiFi AP, and then both have the same bandwidth. To open the same webpage, the smartphone took 30 seconds, while laptop only took 5 seconds.

Power Consumption: The long delay and high CPU usage of web browsing also consume much power of the smartphone. Figure 4 shows the power consumption of opening a webpage and the power consumption of downloading 522KB of data. As can be seen, using web browser will keep the power consumption at 1.08 W for a much longer time compared to downloading the same amount of data (522KB) directly.

The long delay of opening a webpage contributes to significant part of the power consumption. Javascripts and flash are another part of power consumption after the webpage is opened. This is because many of them keep consuming CPU even if the user does not do anything. Figure 5 shows the CPU usage of a web browser on a smartphone after the webpage has been completely opened. Because the web server "alibaba" has a lot of flash and other dynamic web content, the CPU utilization is very high. Although the "dell" webpage looks static, there are javascripts running in the background and they continue to consume CPU. "wikipedia.org" does not have any dynamic content. After being opened, it does not consume any CPU. Although dynamic web content consumes most CPU cycles and battery power on the smartphone, there is little benefit to the user. Simply disabling these dynamic web content may not work since some content would not be shown on the browser.

Although the computing capability of smartphones is expected to increase in the future, the web designer will design more complex webpages which consume more computation power. This motivates our design of the VMP architecture that aims to *shift* computing of web services from smartphones to the cloud computing environment.

IV. Virtual Machine based Proxy (VMP)

In this section, we first give an overview of the VMP system, and then present the detailed design of the communication mechanisms and the VMP.

A. System Overview

To use our VMP, the smartphone has to install an interaction module, which communicates with the proxy. With our VMP, a web request from the smartphone will be sent to the proxy which forwards the request to the web server and gets the reply from it. Then, the proxy sends the processed reply to the smartphone through the wireless network.

As discussed in the last section, to deal with the local computation bottleneck, we let the proxy handle all the http requests, replies, and execute various java scripts and flashes. More specifically, on the proxy side, X11 [17] technique is used to display the webpage on a "virtual" screen rather than a physical one. Then, a screen copy of the current webpage is compressed with zlib [18] and then transferred to the smartphone.

We implement an efficient communication mechanism for saving the power consumption on smartphones. As discussed in Section II, the reason is that the communication mechanism of thin-client approaches is not efficient and consumes too much power on mobile devices. Furthermore, because of the long latency of 3G, the delay of display feedback of user operation is very significant.

In our VMP, by moving the browsing service to the proxy, the users' activity is visible to the proxy. Web access history or any other user-sensitive data is recorded in the browser's cache and other logs. Critical information such as digital certificates is also silently cached. Therefore, it is critical to isolate user information in the proxy to provide privacy and security protection. To achieve this goal, each smartphone uses its dedicated VM on the proxy.

The scalability of our VMP architecture is an important issue. VMP can achieve high scalability through leveraging the *scale-out* functionality provided by many of the current cloud services. In case of Amazon EC2 (Elastic Compute Cloud) [19], the Amazon allows users to define scaling actions and conditions on which to trigger the actions. Windows Azure [20] also provides similar *auto-scaling* features. Well-known businesses such as KBB.com or Pizzahut has been successfully using Windows Azure's auto-scaling to handle excessive burst of workloads. Dynamically varying workloads of the VMP architecture make it an ideal candidate for such *auto-scaling* features to handle almost unlimited scalability.

Figure 6 illustrates the overall architecture of our VMP. We assume Xen-based virtualization environment [21], and it will not limit the generality of the design. The big box represents a physical VM host, which contains a VM and Domain-0. Domain-0 is a specialized administrative VM that is responsible for managing hardware resources. All the I/O of the VMs must go through Domain-0 in the current Xen. Every physical host machine has one Domain-0 instance whereas there can be multiple guest VMs. There are three types of VMs in our design - *master VM*, *Web Cache VM*, and the guest VM. The guest VMs are used for actual shift computing. Inside each VM, the Interaction module is responsible for the interaction between the VM and the client software on the smartphone. The master VM is the connection hub for the smartphone users. Users only need to know the address of this master VM and it assigns a VM. The VMP architecture can reduce the latency and network traffic of the smartphone. Although the data access delay between the VMP and the web server is relatively low compared to the wireless part, there is a motivation to further reduce the traffic and latency between the web server and the VMP through caching; i.e, a web cache VM is added in our VMP architecture so that the VMs can share the web cache.

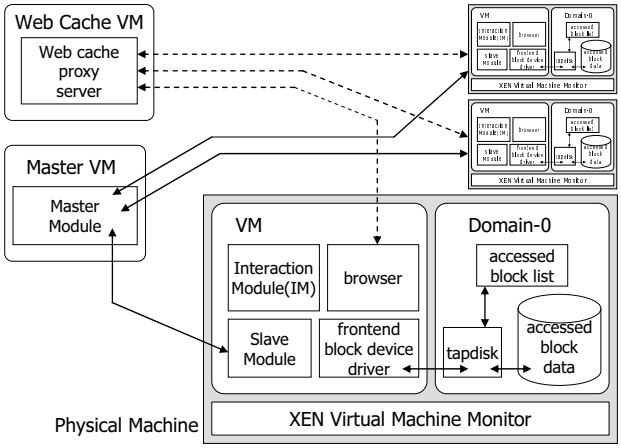

Figure 6. Service-provider side VMP architecture

B. Communication Mechanisms

We develop features to improve the power efficiency of communication mechanism for both opening webpages and user operations after opening them.

Features for opening webpages: Instead of sending the characters of URL one by one, the client only sends one URL request to the server and gets one response which includes the screen display copy from the server. After getting the URL request, the server freezes the display update, until the webpage is opened completely. Otherwise, some temporary incomplete content may be sent to the client who may not be interested in. In addition, most webpages cannot be displayed in one screen due to the small display size of the smartphone. The original approach is to only send the current screen and send the next screen if user requests. However, because of the long latency of 3G, it aggravates the delay experience on the user side, when they switch to the next screen. To eliminate this delay, we develop a multiple screen feature in which the server can send multiple screens of a webpage to the client at once according to the user request.

Features for user operations after opening webpage: To enable the interaction between the user and the proxy,

whenever the user moves the mouse, the client records the mouse position. When the position is on the boundary of the screen, or when the user clicks a link, the client sends this information to the proxy. Then, the proxy maps the mouse position at the client side to the position at the proxy side to open the webpage, and sends the new display screen back. When the user moves the mouse to fill a text field on the webpage, the client records the mouse position and the user input. After the user finishes, all data will be sent to the proxy which sends back the display feedback. To fill out multiple text fields, such as a table, each table content and its position are buffered and sent to the proxy together. To handle display update such as flash, new data is only sent to the interested user. Since the size of the smartphone screen is only one fourth of the screen at the proxy side, the proxy only needs to update that part of the screen. Thus, users still have web flash effect with our approach, but it does not consume any power when users are not interested in it.

C. The VM Management

The master VM serves as a directory server for normal VMs. When the master VM receives a connection request from a new client, it looks up the list of available VMs and assigns one of them to the client. The master VM thus maintains the states of all available VMs. The master VM communicates with other VMs through TCP. Detection of the client's entry into the system is easily done by accepting a request for VM assignment. Detection of client's departure requires a notification from the client or being inactive for a long time. Once the master VM learns that the client has left, it initiates the restart of the VM which can later be used to serve other requests.

Although running the service within a dedicated VM strengthens the security from other currently active VM instances, the user data is still vulnerable to any subsequent user that uses identical VM instance after the current user is finished. We would like to eliminate such vulnerability by resetting the VM image to the initial stage after the user leaves the VM. One easy way to achieve this is to delete the VM images when the client leaves and create a new one. However, because of the large size of the VM image, it takes considerable time to do this. To address this issue, we maintain the list of modified blocks of the VM image while the user was using it. The original values of the modified blocks are appended and kept in a separate file. When the VM shuts down as the user leaves, modified blocks are overwritten back to their original values.

In the implementation we utilize Xen's blktap mechanism [22]. The *tapdisk* process in Xen provides a convenient location to intercept VM-generated block requests and perform some actions. We intercept all the write requests and maintain the list of all the modified blocks with their original value. We have created the accessed block list and accessed block data file. This list is updated whenever the tapdisk process receives write requests. The tapdisk process defines a callback interface for the close event. When the user disconnects, the master VM orders the VM to reboot. As part of the reboot process, we have added a function to restore all the modified blocks to their original values.

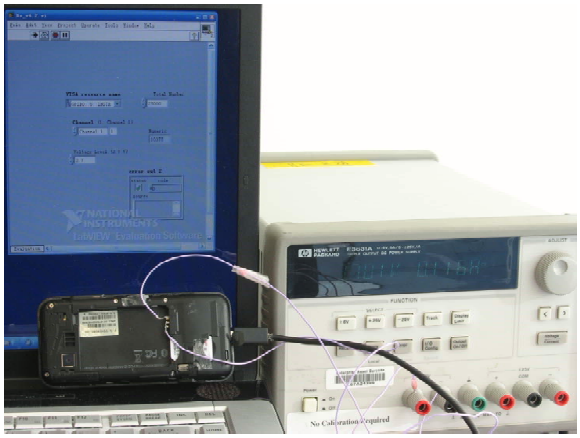

Figure 7. The experimental setup to measure the power consumption of the smartphone

V. Performance Evaluations

In this section, we first describe our experimental setup and our webpage benchmark. Then, we evaluate the delay and power consumption of web browsing.

A. The Experimental Setup

In our experiment, the Android Dev Phone 1 with Android 1.5 is used as a smartphone. It uses T-Mobile 3G UMTS network. We use a Linux PC at the proxy side which runs VMP with Xen 3.1.4. Each VM uses Ubuntu 8.04 Desktop OS. The proxy is connected to the IP network and can be accessed by the smartphone through T-Mobile 3G UMTS network.

Figure 7 shows the experimental setup. To accurately measure the power consumption of the smartphone, we use Agilent E3631A Power Supply to provide the current with constant voltage (3.706 V) to the smartphone instead of using the battery. The Agilent E3631A connects to our laptop through the "NI GPIB-USB-HS" cable. We install LabVIEW on the laptop, and use it to program the Agilent E3631A to capture the current of the smartphone every 0.25 second. Then, we can easily calculate the power consumption of the smartphone.

B. The Webpage Benchmark

Our webpage benchmark consists of 20 popular webpages, which are selected from the top popular webpages listed on the Alexa website [23]. To have a fair comparison between the VMP approach and the normal web browser, the selected webpage should meet the following two requirements. First, it should be correctly rendered by the smartphone browser. We will not use webpages that cannot

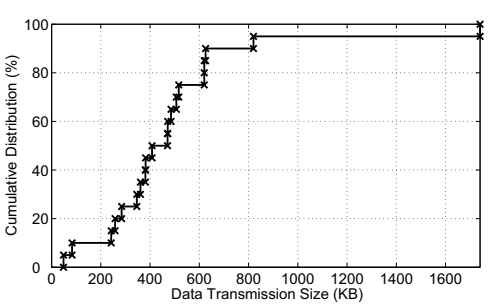

Figure 8. The cumulative distribution of the data transmission size of benchmark webpages

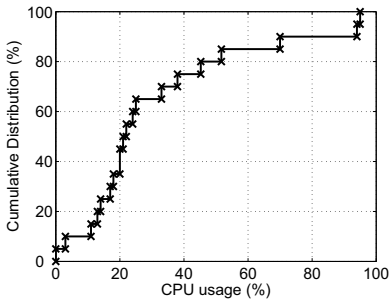

Figure 9. The cumulative distribution of the CPU usage after opening the benchmark webpages

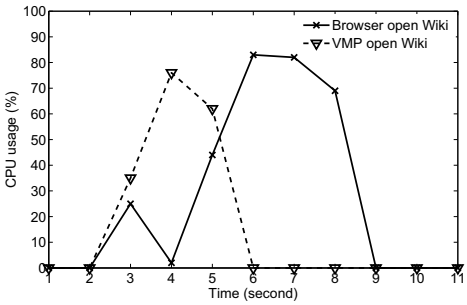

Figure 10. The CPU usage for opening Wiki

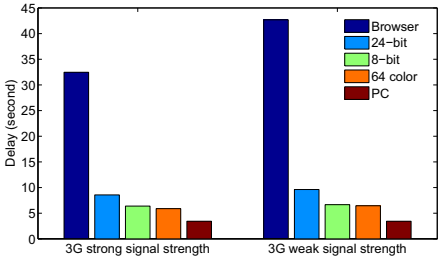

Figure 11. The average delay for opening benchmark webpages

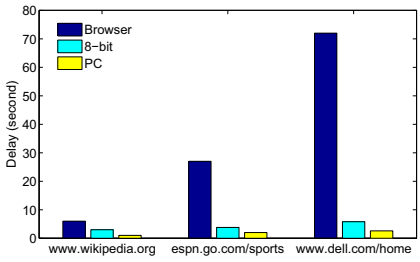

Figure 12. The delay of extreme cases with strong 3G signal

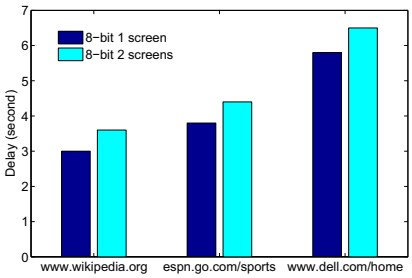

Figure 13. The delay for transmitting multiple pages together

be displayed correctly by the smartphone browser because of lack of software support. Secondly, we will not use the webpage that provides different versions specifically for mobile devices. For these webpages, the actual content will be different when they are opened by the VMP from being opened by mobile devices. The cumulative distributions of the webpage size and the CPU usage after being opened are shown in Figure 8 and Figure 9. The average data size of these web pages is 483 KB, and the average CPU usage after being opened is 31.7%.

We cannot select too many webpages in our benchmark due to the limitation of the cache size of the Android browser. The maximum cache size is only 8MB. To evaluate the performance of the web browser with cache, we have to limit the number of webpages to make sure that there is enough cache space to hold them. We will not use existing benchmarks such as SPECweb2009 [24] because they are developed for evaluating web server performances or because they are too old to be useful as is the case for the i-Bench [25] which was developed in 2000.

C. Experimental Results on Delay

In this section, we first present the average delay for opening benchmark webpages, and then explain the reason behind it. We also evaluate the effects of compression and transmitting multiple screen copies on delay.

1) The average delay for opening benchmark webpages: In this section, we evaluate how much delay the VMP approach can reduce, compared to the normal smartphone browser. In VMP, the delay for opening a webpage consists of three parts: the delay to send the request to the proxy, the delay to open the webpage at the proxy, and the delay to receive the data from the proxy. When the proxy sends the screen copy of the webpage to the smartphone, there are different resolutions to choose: 64 color, 8-bit (256 color), 16-bit, 24-bit.

Figure 11 shows the average delay of accessing the benchmark webpages. It compares five cases: the normal smartphone browser, the normal PC, and three cases of VMP: 64 color, 8-bit, and 24-bit. These comparisons are under two different environments: strong signal strength and weak signal strength. To get strong signal, we use the smartphone near the base station. For weak signal, we use the smartphone in our office where the signal strength indicator shows minimal value. Because the signal strength affects the data transmission rate, it generally takes longer time under weak signal to download the same amount of data.

As shown in Figure 11, using PC to access the web has the lowest delay, using smartphone browser has the highest delay, and the VMP approach is in the middle. Comparing three VMP approaches, the 64 color approach has lower delay than the 8-bit approach which is lower than the 24-bit approach. Intuitively, the resolution of 24-bit will be much clear than the 64 color, but it needs to transmit much more data. Note that the delay of VMP includes the delay for the proxy to open the webpage which is almost equal to the delay of the PC approach. After removing this part of delay, the delay of the 24-bit approach is much higher than that of the 64-color approach due to transmitting much more data. Since the smartphone screen is very small, the difference of different approaches is not that clear, and the 8-bit approach

Table I
OPEN WEBPAGES WITH VMP AND BROWSER

	VMP	Browser
Wiki delay (s)	3	6
nytimes delay (s)	5.9	41
Wiki data transmission (KB)	109	83.3
nytimes data transmission (KB)	493.5	522

looks good. Therefore, we choose the 8-bit resolution for our VMP. From the figure, we can see that our VMP (8-bit) approach can reduce the delay by 80% compared to normal smartphone browser.

2) The delay of extreme cases: To understand the long delay for web browsing, we need to look into the details. We study two cases: one is to access "www.wikipedia.org" which has 83.3KB of data, and the other one is to access "www.nytimes.com" which has 522KB of data. Table I shows the delay and data transmission size for opening these two webpages with VMP and smartphone browser.

As can be seen from the table, using the smartphone browser has much longer delay when accessing nytimes compared to VMP. As explained in Section III, the CPU usage of opening nytimes with smartphone browser reaches almost 100% most of time. It also shows that the local computation related to web browsing is the bottleneck, which is the key reason for the long delay in web browsing. On the other hand, the VMP approach does not have the local computation bottleneck, and hence it has much lower delay to open the webpage.

From Table I, we can also see that the delay of accessing "www.wikipedia.org" which is very simple and small, is still longer than the VMP approach. This can be explained by Figure 10 which shows the CPU usage for opening wiki by smartphone browser and by VMP. As shown in the figure, at time 3, the button is clicked to open the webpage, and some CPU time is used to handle the screen touch action. The smartphone browser uses much more CPU power to open the webpage. It takes 6 seconds to open the webpage which include time to execute various stages of the http protocol and handle some local computation. The VMP approach only takes 3 seconds to open the webpage.

We also measure the delay of accessing some typical webpages. Figure 12 shows the results of accessing three webpages: "www.wikipedia.org" , "espn.go.com/sports" , and "www.dell.com/home". In small webpage such as "www. wikipedia.org", VMP only has a small advantage compared to the normal smartphone browser. However, for webpages which have many javasciprts and flashes, VMP reduces the delay by 86% in opening "espn.go.com/sports", and reduces the delay by 92% in opening"www.dell.com/home". As explained earlier, the VMP approach shifts the computation to the proxy, and hence reduces the delay.

3) The effects of transmitting multiple screens of data: Since most webpages cannot be displayed on one screen, we evaluate the delay of transmitting multiple screens of data to the smartphone. Figure 13 shows that compared to the delay of transmitting one screen of data, the delay for transmitting two screens of data only slightly increases about $0.6 \sim 0.8$ seconds. In VMP, transmitting multiple screens of data only increases the delay to receive the data from the proxy. The data size for transmitting an additional screen of data is only about 70 KB. Because the smartphone bandwidth with 3G connection is around 100 KB/s, the delay increase for transmitting multiple screens of data is very small.

Table II
OPEN WIKI WITH AND WITHOUT COMPRESSION

	with compression	without compression
delay (s)	3	16
data transmission (KB)	109	1460

Table III
THE AVERAGE DELAY OF OPENING BENCHMARK WEBPAGES WITH ANDROID AND IPHONE

	Android	iPhone
Browser without cache (s)	32.465	39.1
Browser with cache (s)	26	29.85
CPU (MHz)	528	620
Memory (MB)	192	128

4) The effects of data compression : VMP uses zlib to compress the display data at the proxy side to reduce the data transmission size. However, the client has to spend a little bit more power on computing to decompress the data. Thus, we evaluate the delay with and without data compression to find out if we also need remove this computing. Table II shows that even for a small webpage such as wiki, without data compression, the data size increases from 109KB to 1460KB, which adds about 13 seconds of delay for a 100 KB/s link. Since the screen copy has lots of redundancy, the data size can be significantly reduced by compression. Also, the decompression function is very simple, and it does not consume too much CPU time on the smartphone. Thus, it is easy to see the advantage of keeping decompression computing on smartphone.

Finally, we want to investigate if the high delay of Android browser is caused by device specific design and implementation. We use iPhone 2.0 to open the same benchmark webpages and measure the average delay with strong signal. Table III shows that the average delay of Android is slightly less than that of iPhone for opening the same benchmark webpage with similar hardware configuration. Therefore, for most 3G smartphones, the high delay of web browsing may be independent of their browser design and implementation, and it is due to the limited computing capability of the smartphone. Therefore, shifting the computing away from the smartphone is a good solution to address the limited computation capability and to reduce the delay and power consumption of the smartphone.

D. Experimental Results on Power Consumption

To understand the power consumption of smartphones, we have performed several experiments and we present the

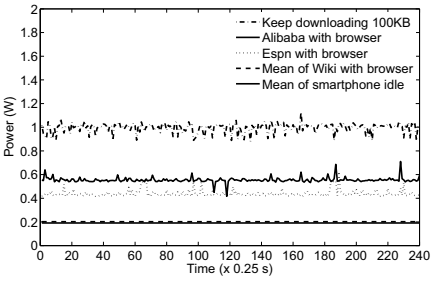

Figure 14. The power consumption of typical webpages after being opened

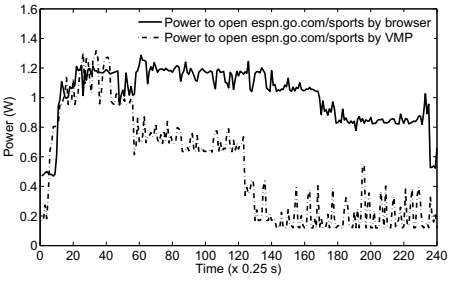

Figure 15. The power consumption for opening ESPN sport

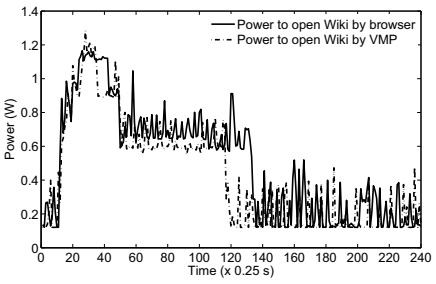

Figure 16. The power consumption for opening Wiki

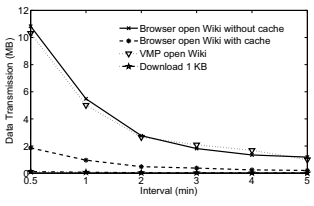

Figure 17. The amount of data transmitted when repeatedly open Wiki with different time interval for an hour

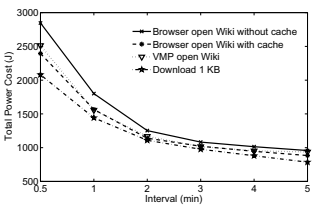

Figure 18. The amount of power consumed when repeatedly open Wiki with different time interval for an hour

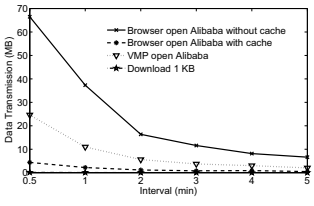

Figure 19. The amount of data transmitted when repeatedly open Alibaba with different time interval for an hour

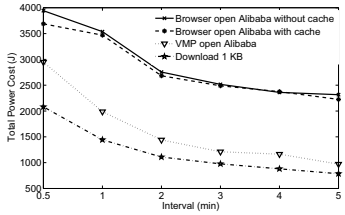

Figure 20. The amount of power consumed when repeatedly open Alibaba with different time interval for an hour

results in this subsection.

1) Power consumption of smartphone browser after being opened: In this experiment, we show the power consumption of the smartphone browser after being opened. We use the following three typical webpages.

- Wikipedia.org: small size webpage; needs to transmit 83KB data; will not consume any CPU power after being opened.
- espn.go.com/sports: medium size webpage; needs to transmit 421KB data; keeps using 19% CPU power after being opened.
- alibaba.com: large size webpage; needs to transmit 514KB data; keeps using 100% CPU power after being opened.

In addition to these three cases, we also add the results with two other extreme cases for comparison: smartphone in *idle* state, and in heavy downloading case. Note that *idle* state is different from the *sleep* state. For the heavy downloading case, we use "keep downloading 100KB" described in Section III.

As shown in Figure 14, data transmission consumes the largest amount of power, which is about 1 W. After downloading the website "alibaba", the local CPU is 100% saturated, and it consumes about 0.6 W. For "ESPN", which consumes about 19% CPU after the webpage is opened, the power consumption is about 0.44 W. The power consumption of "wiki" and idle has the lowest power consumption, which is about 0.2W. From this figure, we can see that repeatedly downloading data consumes the most amount of power, and keeping the CPU running also consumes a lot of power compared to keeping it idle.

2) Power consumption of opening webpages: Figure 15 and Figure 16 compares the power consumption between VMP and normal smartphone browser when opening webpages. For opening "espn.go.com/sports", the power consumption of VMP has been dropped at time 60, since it does not need to transmit any data after that. But the browser only finishes downloading data at time 160. For opening "wiki", which only has a small amount of data. The power consumption of both approaches has similar pattern. The only difference is that VMP finishes downloading a little earlier as explained before (see Figure 10). So the power consumption of VMP drops at time 120 whereas that of the browser drops at time 140.

3) Power consumption in extreme cases: In this section, we evaluate the power consumption of VMP and browser for two extreme cases: "wiki" (CPU usage 0%) and "alibaba" (CPU usage 100%). We repeatly open the same webpage with certain interval for an hour to get an idea of the power consumption in a more general setting. We have two cases "with cache" and "without cache". To get the results of "without cache", the web cache and cookie of the browser are erased before being opened next time. This is achieved by modifying the browser of Android to automatically do the test and erase the cache before each opening. To compare with the browser approach fairly, VMP send all screen copies of the webpage to the smartphone.

Figure 17 compares the amount of data transmitted in the following four cases: browser with cache, browser without cache, VMP, and repeatedly downloading 1KB of data. This 1KB case will give a lower bound on the amount of data transmitted. The 1KB case transmits the smallest amount of data, and the browser with cache has the next lowest amount of data. The other two have similar amount of data. As the time interval increases, the amount of transmitted data decreases since the phone spends more time in the idle

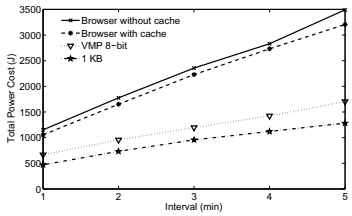

Figure 21. Power consumption with strong signal

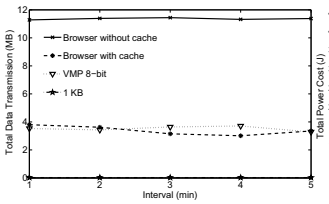

Figure 22. Data transmission with strong signal

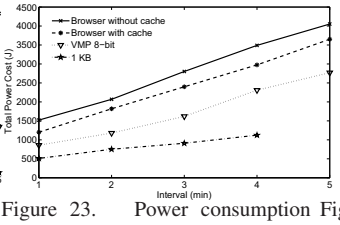

Figure 23. Power consumption with weak signal

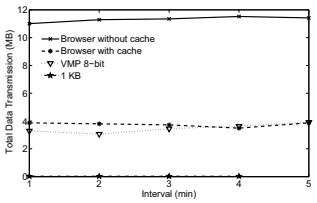

Figure 24. Data transmission with weak signal

stage.

Figure 18 compares the power consumption of these four cases. As expected, the 1KB case provides the lower bound for power consumption. However, the power consumption of the four cases is similar although they transmit different amount of data. The power consumption is mainly decided by the time to open it. As shown in Table I, VMP needs about 3 seconds open wiki, similar to "wiki with cache". The delay of "wiki without cache" is about 6 seconds. This explains why the VMP approach has lower power consumption than "wiki without cache".

Figure 20 compares the power consumption of these four cases when opening webpage "alibaba". This time, when the interval is 0.5 minutes, VMP approach cuts the power consumption by 25% compared to the browser approach. As explained before, the power consumption is related to the delay to open it. The delay to open "alibaba" by the browser without cache is 35 seconds, while the delay with cache is 24 seconds. The delay to open it by VMP is 7 seconds. Although "alibaba" with cache downloads much less data compared to VMP (as shown in Figure 19), it still consumes much more power because the delay to get the data is much longer than VMP due to heavy local computation.

As the time interval increases to 5 minutes, the data downloading time becomes much smaller compared to the idle time. Thus, the power consumption between VMP and the 1KB case becomes closer since both have the same power consumption during idle. The VMP approach reduces the power consumption by about 58% compared to the browser approach. This is because the power consumption is mainly decided by the CPU utilization when the time interval is long. Since "alibaba" still utilizes 100% CPU after the browser opens it, its power consumption is much higher than VMP which is idle most of the time. Note that there is no difference between the "cache" and "without cache" since both need 100% CPU.

4) Power consumption with the webpage Benchmark: In this section, we use the benchmark webpages to compare the power consumption of those four cases. In this experiment, instead of opening the same webpage, the smartphone opens each webpage in the benchmark and stays for certain time interval, then opens another webpage until all 20 webpages have been opened.

Figure 21 compares the power consumption under strong signal, and Figure 22 compares the amount of data transmitted. Since only 20 webpages will be open, the amount of transmitted data does not change as the time interval increases. However, the amount of power consumed will be increased as the time interval increases because the phone will spend more time in the idle stage which also consumes power.

The results are generally consistent with the results in Section V-D3. Although "Browser with cache" can reduce the amount of data transmitted, the power consumption is still close to "Browser without cache" because it takes close amount of computing to open the webpage and still takes same CPU utilization after the webpage is opened. However, compared to using the browser, VMP can reduce the power consumption by about 45%, and it is close to the lower bound provided by the 1 KB case.

Figure 23 and Figure 24 compares the power consumption and the amount of data transmitted when opening the benchmark webpages under weak signal. The major difference between these results from those under strong signal is that more power is consumed under weak signal. This is because weak signal will reduce the data transmission rate and increase the transmission time, and hence consume more power. When the signal strength is very weak, if the client does not have any data transmission with the server for a long time, the connection will be lost. That is why we cannot obtain the power consumption for the "1 KB case" with 5 min interval.

VI. Discussions

There are different ways to deploy the VMP system. For example, the service provider such as T-Mobile, AT&T or Verizon can use the existing public cloud computing service to build the VMP system. Although this approach has low cost, public cloud may not meet the security level, and the network traffic between the cloud and the 3G network may introduce extra overhead. It is better for these service providers to build their own private cloud, which can be located closer to the 3G infrastructure to reduce the network traffic. Also, the service provider can provide other value added service later if they have the private cloud. There are other ways to adopt the VMP architecture at the initial stage. For example, some early users can run the proxy on their office/home computers. An organization can run a proxy to serve employees inside that organization or paid users. Building VMP on top of cloud computing mostly

takes care of the capacity issue because one of the main benefits of cloud is the ability to automatically scale based on the workload.

There are some limitations with our prototype related to security and privacy [26], [27]. Although the VM based approach can address most security and privacy issues, some applications such as banking access may require more security and privacy. Then, we have to encrypt the message between the phone and the proxy. There are also issues such as whether the user will trust the proxy or not. If not, we may rely on techniques such as trusted computing module to make sure the information at the proxy cannot be used for other purpose. Since most people may not use smartphone for these high secure applications or they can switch to the normal browser when using these applications, we leave this part for future work.

Another benefit of this VMP architecture is that we can make the smartphone more secure. Since many malwares come from the web, letting the proxy first open the webpage can address many security issues. Since the proxy is more powerful, it can run CPU intensive anti-virus software and identify the problem earlier.

VII. Conclusions

In this paper, we have discovered that the key reason of the long delay and high power consumption in web browsing is not due to the bandwidth limitation most of time in 3G networks. The local computation limitation at the smartphone is the real bottleneck for opening most webpages, especially for those with rich dynamic content such as javascripts and flash. To address this issue, we have designed an architecture, called Virtual-Machine based Proxy (VMP), to shift the computing from smartphones to the VMP. With our VMP, a web request from the smartphone will be sent to the proxy which forwards the request to the web server and gets reply from it. Then, the proxy sends the reply to the smartphone through the wireless network. Instead of sending the original HTTP reply which contains many CPU intensive objects, the proxy simply sends a processed screen copy to the smartphone. Experimental results show that compared to normal smartphone browser, our VMP approach reduces the delay by more than 80% and reduces the power consumption during web browsing by more than 45%.

References

[1] L. Zhong and N. K. Jha, "Energy efficiency of handheld computer interfaces: limits, characterization and practice," in *Proc. ACM MobiSys*, 2005.

[2] J. Flinn and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Transactions on Computer Systems (TOCS)*, May 2004.

[3] J. Liu and L. Zhong, "Micro power management of active 802.11 interfaces," in *Proc. ACM MobiSys*, 2008.

[4] 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*, June 1998.

[5] T. W. Bickmore and B. N. Schilit, "Digestor: device-independent access to the World Wide Web," in *Proc. International World-Wide Web Conference (WWW)*, 1997.

[6] A. Moshchuk, S. D. Gribble, and H. M. Levy, "Flashproxy: transparently enabling rich web content via remote execution," in *Proc. ACM MobiSys*, 2008.

[7] (2010) Opera mini. [Online]. Available: http://www.opera.com/mobile/features/

[8] B.-G. Chun and P. Maniatis, "Augmented smart phone applications through clone cloud execution," in *Proc. HotOS XII*, 2009.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, 2009.

[10] E. Cuervoy, A. Balasubramanianz, D.-k. Cho, A. Wolmanx, S. Saroiux, R. Chandrax, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. ACM MobiSys*, 2010.

[11] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proc. ACM MobiSys*, 2003.

[12] T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper, "Virtual Network Computing," *IEEE Internet Computing*,Jan 1998

[13] J. Kim, R. A. Baratto, and J. Nieh, "pTHINC: a thin-client architecture for mobile wireless web," in *Proc. WWW*, 2006.

[14] J. Smith, R. Mohan, and C.-S. Li, "Content-based transcoding of images in the internet," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 1998.

[15] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

[16] A. B. King. "Average web page size triples since 2003," http://www.websiteoptimization.com/speed/tweak/average-web-page/, 2008.

[17] T. Richardson, F. Bennett, G. Mapp, and A. Hopper, "Teleporting in an x window system environment," *IEEE Personal Communications*, September 1994.

[18] P. Deutsch, "RFC1950: ZLIB compressed data format specification," *IETF*, May 1996.

[19] "Amazon Elastic Compute Cloud (EC2)," http://www.amazon.com/ec2/.

[20] Microsoft Azure, http://www.microsoft.com/azure/.

[21] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. ACM SOSP*, 2003.

[22] TimBenke, "Xen blktap," http://wiki.xensource.com/xenwiki/blktap, March 2008.

[23] Alexa Internet, Inc, "The top 500 sites on the web," http://www.alexa.com/topsites, December 2009.

[24] SPECweb2009, "Standard Performance Evaluation Corporation," http://www.spec.org/web2009/, June 2009.

[25] Ziff-Davis, Inc, "i-Bench version 1.5," http://etestinglabs.com/benchmarks/i-bench/i bench.asp, 2000.

[26] B. Zhao, C. Chi, W. Gao, S. Zhu, and G. Cao, "A chain reaction DoS attack on 3G networks: Analysis and defenses," in *Proc. IEEE INFOCOM*, 2009.

[27] Z. Zhu and G. Cao, "APPLAUS: A privacy-preserving location proof updating system for location-based services," in *Proc. IEEE INFOCOM*, 2011.