Modeling Mobile Web Characteristics for Energy Optimized Delivery

Troy Johnson

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

Department of Computer Science

Central Michigan University
Mount Pleasant, Michigan
September 2014

ACKNOWLEDGMENTS

This work was sponsored by an Early Career Grant from the Office of Research and Sponsored Programs at Central Michigan University. I wish thanks to A. Sampson and R. Kohvakka for assistance in the development of the testbed measurement environment.

ABSTRACT

Modeling Mobile Web Characteristics for Energy Optimized Delivery

by Troy Johnson

As mobile traffic and data consumption continue to rise, there is a growing need to investigate increased energy efficiency and optimizations to reduce the bandwidth when browsing the mobile web. Use cisco figure citations here???? To determine the reduction in energy consumption of mobile devices, there is also a need for a way to measure the energy consumption of mobile devices. By investigating the composition and characteristics of mobile web pages, statistical models can be derived for describing the characteristics of a typical mobile web page, such as the individual response sizes and expiration ages of responses that mobile browsers request for web pages.HTTP Archive will be a great source of data that may be utilized to derive models for describing the mobile web. Additionally, this pool of data is updated on a bimonthly basis, providing a constantly updated pool of data to update the developed models with and validate them. These models can then in turn be used to provide more accurate results when estimating the possible energy and bandwidth savings by using these models for generating artificial web pages that will contain characteristics that closely resemble those characteristics often found on the actual mobile web. Investigating the models and data further, they can be extended to create prediction models that will describe the growing mobile web for future years. With these models in place, they can be applied to projects for optimizing energy and bandwidth consumption on mobile devices, such as the possible energy and bandwidth savings that can result from cache forwarding between desktop computers and mobile devices. To measure the possible energy consumption savings from these projects, a low cost test bed for measuring the power consumption of mobile devices can be employed as a baseline

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

HTTP Archive

HTTP Archive is an online repository of web performance information containing information on both desktop and mobile versions of websites. Information gather includes all the details about the responses each webpage makes such as the response sizes, expiration age, HTTP Archive gathers their data using a private instance of WebPageTest [CITATION????].

CHAPTER II

An Inexpensive Testbed for Mobile Device Power Measurement

Introduction

Over the last few decades, there has been an enormous increase in the ubiquity of mobile devices. With this increase, has also come the increase in demand for data-driven services and this demand is predicted to continue [1]. The battery consumption of mobile devices represents a limiting bottleneck and thus power optimizations suggestions have been suggested [2]. Software-based energy profilers do exist [3], however they are not always feasible for implementing in a straightforward manner or desirable due to rapid development cycles. To overcome these barriers, a real world test bed that can be implemented which to perform measurement of power consumptions on mobile devices.

Hardware Configuration

The main component in this testbed is the mobile device. This can be realized by using a smartphone and replacing the battery with connectors to the power supply; alternatively one of the common development board packages, such as Pandaboard (see www.pandaboard.org) or Wandboard (see www.wandboard.org) packages. Development boards and smartphones were utilized together with the Android operating system, which provides log output via USB to the measurement control device, which can be a regular PC or another development board with Android debugging support. The mobile device is then networked with a wireless access point, which allows for wired and wireless evaluations. The switchable power supply has an external serial or USB port to communicate the current and power in small time intervals to the control device. A BK Precision 1696 switchable power supply is utilized, as it offers fine granularity in power, current, and time intervals. While other equipment, such as Arduino with custom circuits, were used in other measurement approaches, these

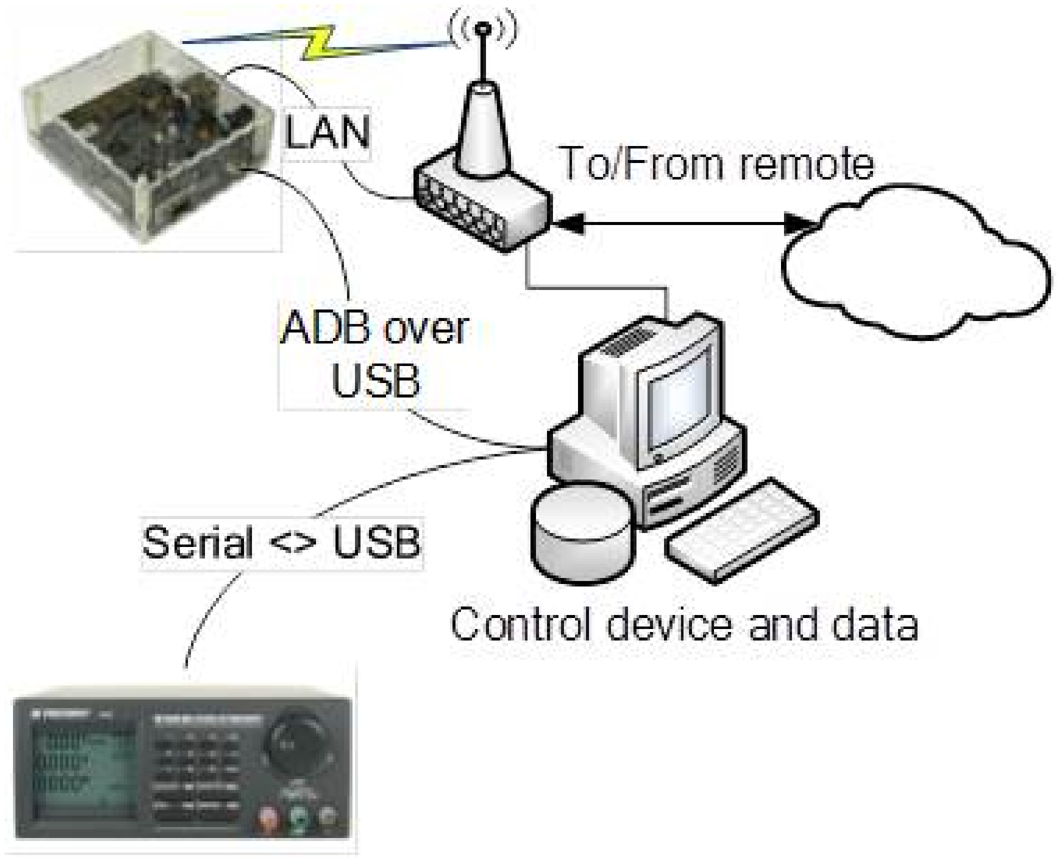


Figure 1. Illustration of measurement testbed.

power supplies are common lab equipment and offer overall robust features.

Software Configuration

The software components are comprised of several Python scripts that execute the Android Debugging Bridge (ADB) and capture the output either to a local file or allow sending the output to a remote receiver, as illustrated in Figure 1. The scripts allow for easy customization on the locally connected control device or at a remote location, e.g., filtering by specific events in the log. Similarly, a locally executing script captures the output from the power supply and is enabled to forward the data to a remote location as well.

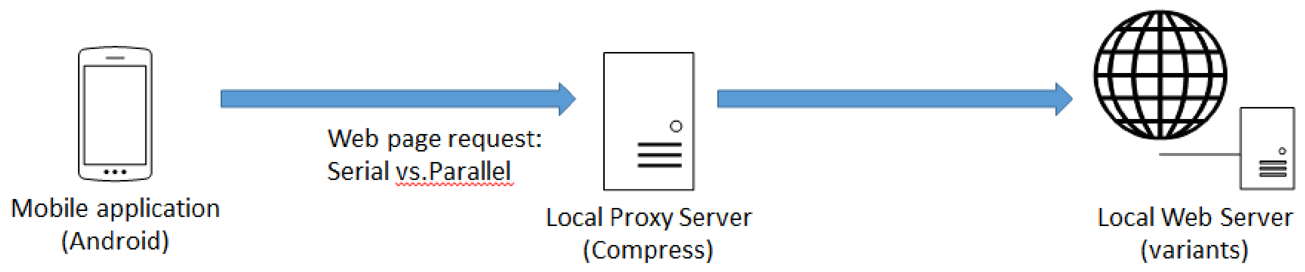


Figure 2. Workflow of mobile application tested on testbed.

Demonstration Description

To demonstrate the usefulness of the testbed, two different aspects of the measurement setup were utilized using an example Android application that performs web requests. This application makes requests to a local proxy server either serially or in parallel and the proxy server goes out and fetches what the phone requested. The workflow for the application ca be seen in figure 2. Both, wired and wireless access scenarios were exhibited for accessing a remote web service and retrieving results in order to demonstrate the functionality of the testbed. With these demonstrations, real time visualizations of the data about power consumption were created and stored on log files on a remote computer where they can be readily parsed for automatic evaluation of application power consumption.

CHAPTER III

Power Consumption Overhead for Proxy Services on Mobile Device Platforms

Introduction

Current predictions by Cisco show that the amount of data that users consume has increased significantly and will continue to increase into the foreseeable future [1]. Previous studies show that the network interfaces of mobile devices consume much of the limited battery life [4]. Thus, heavy research efforts have been poured into studying the possibilities of energy efficient mobile data delivery. Some research avenues have middleware that acts as an on-device proxy service to realize benefits or enable new interaction paradigms, such as display networks [5] or mobile content sharing [6],[7]. To determine whether or not local proxy servers result in a large overhead in terms of power consumption and time delays, the measurement framework testbed described in Chapter II can be implemented to determine what kind of overheads can be expected from local proxy servers.

Methodology and Metrics

Measurement Setup

At the core of the setup, a Pandaboard ES mobile software development platform is utilized, which features a Texas Instruments OMAP 4460 dual core ARM Cortex-A9 processor with 1 GB of DDR2 RAM, SMSC 10/100 Mbps Ethernet port, and LS Research WLAN/Bluetooth wireless module, next to other components. (Please refer to http://www.pandaboard.org for more details. The open-source Android distribution (version 4.1.2, âĂŸJelly BeanâĂŹ) is used as the operating system software for the mobile device. The overall measurement setup in can be seen in Figure 3. The Pandaboard is powered by a BK Precision 1696 switchable power supply, which features serial port access to read out voltage and ampere values over time. The power supply is connected to a Linux desktop computer serial port which timestamps the values obtained over time to measure the power consumption incurred by the Pandaboard. The Pandaboard is connected through a 1 Gbps maximum speed Ethernet campus network, which eliminates potential bottlenecks. For wireless measurements, an externally connected WLAN antenna is utilized to connect to the campus network through a dedicated WLAN access point, again eliminating bottlenecks for the amounts of data considered throughout. It's also important to note that a combination of input devices and an external monitor were connected as well. On the server side, a locally hosted virtual machine next to Internet-routed web requests is utilized. The local server employs the Debian Linux distribution as its' operating system with the Apache2 HTTP server and the popular Video Lan Client (VLC) as media streaming application.A preencoded video sequence of the popular open-source movie Tears of Steel (see http://www.tearsofsteel.org for more information) is streamed utilizing HTML5 video streaming. The video-only sequence was transcoded offline into a resolution of 864 ÃŮ 480 at 24 frames per second in the Theora video codec and encapsulated using the OGG container format, both commonly utilized for HTTP video streaming âĂIJon the webâĂİ and suitable for mobile playout. The resulting video bit stream has a duration of 12 minutes, 14 seconds and an average bit rate of 1.42 Mbps. The bit rate in turn falls well within range of the network bandwidth capacity.

Mobile SOCKSv5 Proxy Server

Several implementations of the SOCKSv5 standard exist to date [8], which allow utilization of a remotely hosted standard-conforming SOCKSv5 proxy server (typically from a desktop computer through an organizational server). Mobile implementations, however, are less frequent. One example of an implementation for the Android operating system is the anonymity generating Orbot application (see http://www. torproject.us for more details), which routes traffic into the TOR network and contains âĂIJproxificationâĂİ methods for applications as well (i.e., transparently forcing the usage of the proxy through, e.g., modifications of the iptables firewall). A basic Android service application was generated that is based on the jSOCKS proxy server implementation [9], which is open source (entirely written in JAVA) and does not require any privileges, such as root level system access. As the service is executed within the Dalvik VM utilized on Android devices, it incurs a minimal computational overhead when compared to native applications. This approach, however, is commonplace to allow broadest application compatibility and encouraged for developers of the platform.

Performance Metrics

In the following, we briefly outline the metrics used to evaluate the performance of either scheme. Initially, we capture the reported voltage level v(tl) [V] and the current i(tl) [A] as reported by the power supply and timestamped at time tl on the connected desktop computer. We similarly calculate the instant power consumption as p(tl) = v(tl) Âů i(tl) [W]. As the reported values are instantaneous snapshots in time from l = 0 at t(l) = 0 (denoting the first measurement) to l = L, which happened at t(l) = T (whereby T denotes the last measurement), we calculate the time passed between consecutive measurement instances as t(l) = tl âĹŠ tlâĹŠ1. To determine the energy that was used in the l-th measurement period, we calculate e(l) = t(l) Âů w(tl) [J]. We denote the energy that was used in a measurement period up to t(l) as

Need to add in equations.

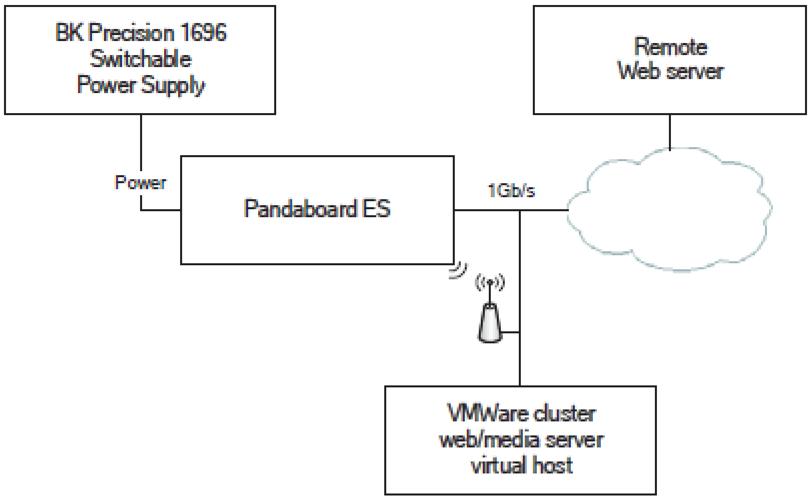


Figure 3. Overview of the measurement setup

Performance Evaluation For Web Request

To perform a representative evaluation of frequent HTTP web requests, web requests for GoogleâĂŹs home page were utilized. The goal of this particular measurement scenario is to evaluate the performance impact of frequent requests through the local proxy server, which has to perform the additional connection tasks each time a request is made. A direct measurement application for Android was developed, which will request http://www.google.com without further resolving any HTTP objects within. Individual requests are followed by a sleep period of 2 seconds for both, direct requests and requests through the mobile SOCKSv5 proxy service. As requests are made without utilizing a browser, no caching is involved client-side.

Fixed Network

In the fixed network scenario, the Pandaboard is connected through wired Ethernet to the campus network while performing the web requests. The requests typically coincide with high power consumption levels, as illustrated in Figure 4 for an exemplary 100 web requests with both configurations. We observe that both approaches exhibit an initial âĂIJspikeâĂİ behavior and an otherwise low level (with some general noise due to overall device activities). There is no immediately visible trend for the momentary power consumptions, as in both approaches, there are several bursty periods of slightly elevated consumptions on top of the actual web requests. Next, we evaluate the total (compounded) energy consumption that is observed when performing these requests for a certain period of time. We illustrate 100 subsequent requests directly and through the mobile proxy server in Figure 5. Initially, it is observed that despite the short-term fluctuations, there is a linear increase in the energy consumed while using either approach. More significantly, there is no immediately noticeable significant difference between the approaches, which is indicated by the almost indistinguishable values in the plot. Lastly, comparing the overhead between the approaches numerically in Table III.1, where the 300 highest levels of energy consumption measured for periods of placing 300 requests were analyzed. It's also notable that the direct approach results in a higher average level of energy consumption (albeit with a larger variability), whereas the proxy-based approach yields a lower average and variability of energy usage values determined. Overall, this results in an overhead of o = âĹŠ0.0563, which presents an initially counter-intuitive result. (Differently worded, by utilizing the mobile proxy server consuming additional CPU cycles, potential energy savings of 5.6% could be realized without requiring any additional modifications.) Taking into account that partially significant variability exists due to some outliers in the total duration (as some measurement points can exhibit significant delays or coincide with other unrelated system activities), both approaches are very similar.

Interface	Approach	Average[J]	Standard Deviation [J]	Confidence Interval (99 %)
LAN	Direct	0.0912	0.0139	0.0021
LAN	Proxy	0.0860	0.0062	0.0009
WLAN	Direct	0.0900	0.0125	0.0019
WLAN	Proxy	0.0902	0.0089	0.0013

Table III.1. Summary values for 300 direct and proxy-routed web requests over traditional ethernet and wireless LAN networks.

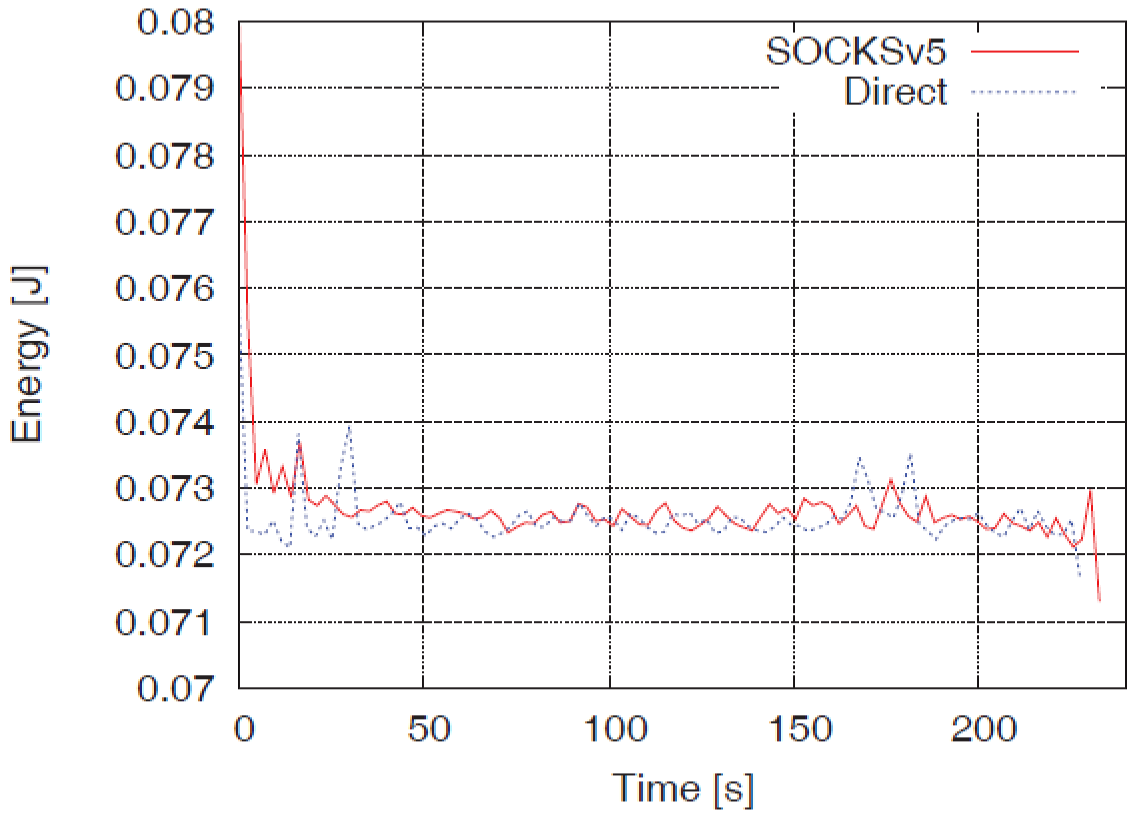


Figure 4. Energy consumption while performing 100 web requests directly or through a mobile SOCKSv5 service, smoothed over time.

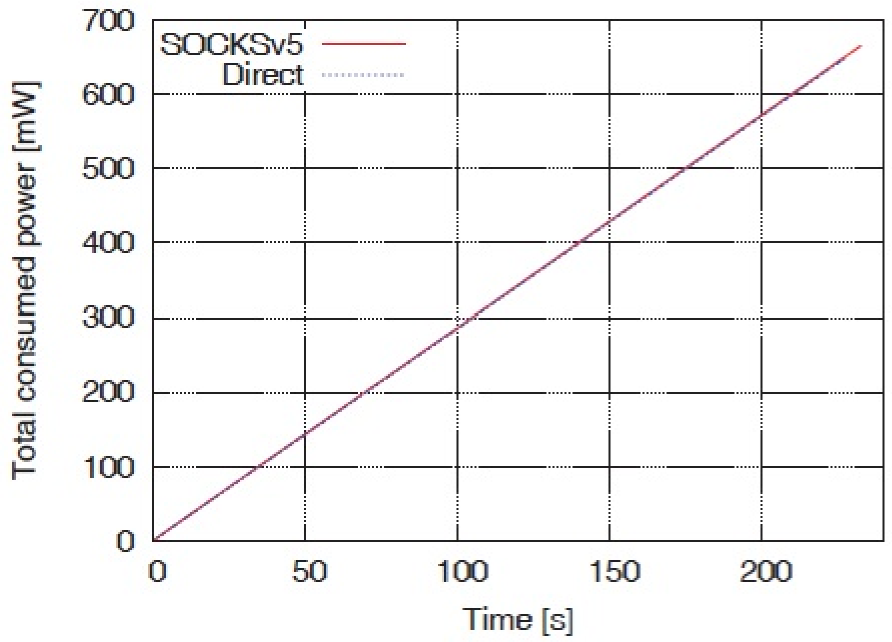


Figure 5. Compounded energy consumption for performing 100 web requests directly or through a mobile SOCKSv5 service.

Wireless Network

Shifting the evaluation to HTTP requests made over the wireless network interface, we present our results in Table III.1. (We note that a graphical evaluation would yield results similar to those presented in Figures 4 and 5.) We initially note that both approaches are very close with respect to their measured average energy consumption, resulting standard deviations, and narrow confidence intervals.

Comparing these results with those obtained for the Ethernet scenario, which outlines the base case without active wireless communications overheads, we do not note a significant difference in the average energy usage for the individual web requests.

Impact of Web Request Variations

Motivated by the closeness of requests, the next step is to more closely evaluate the impact of the web request size over a wireless LAN on the overall power consumption. To limit the impact that external networks can have (such as different delays), the direct performance comparison is performed within the on-campus VM environment illustrated in Figure 3. A dedicated virtual machine uses the Apache2 web server and hosts a Python script that generates a requested number of bytes, additionally eliminating potential caching impacts. 100 repeated measurements are performed for each different web request size and significant outliers are deleted.

Power Consumption

The average energy used per different request size is illustrated in in Figure 6. It's observed that the mobile SOCKSv5 proxy approach always incurs a penalty over the direct connection, which is readily explained by the additional local processing overhead on the device. Furthermore, it can be

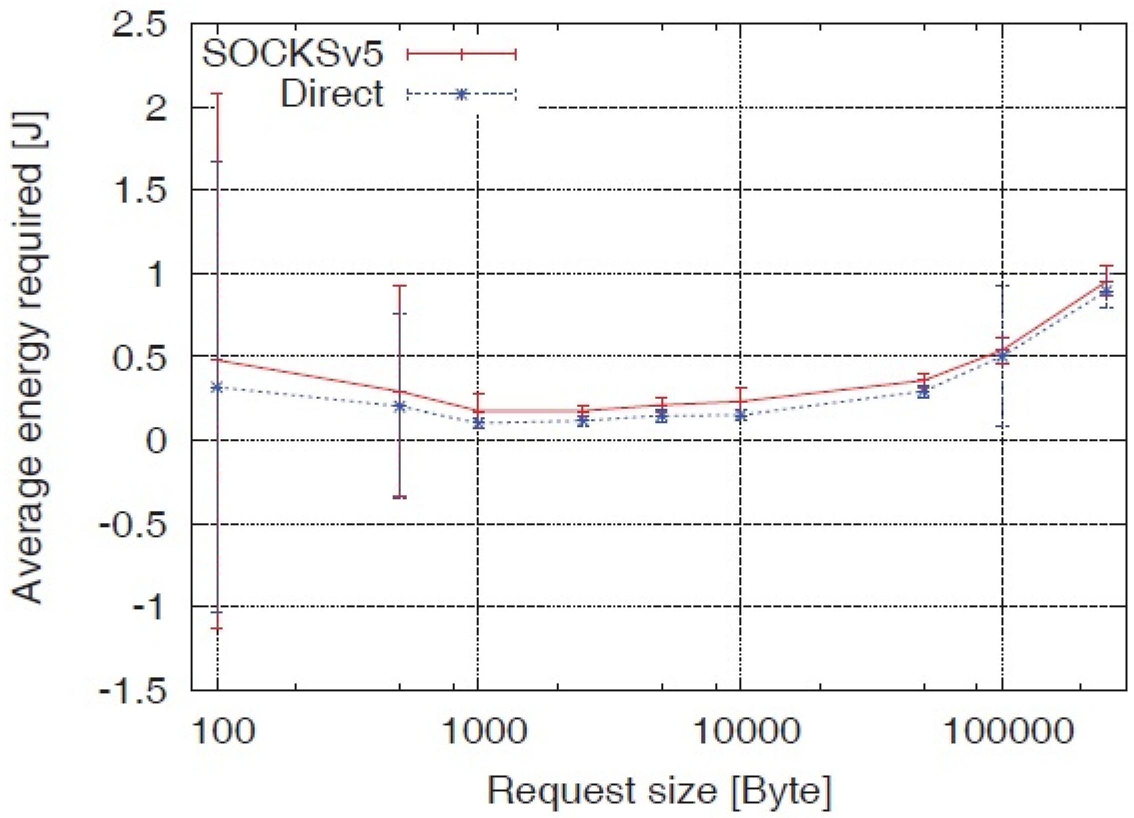


Figure 6. Average energy consumption and standard deviation for requesting different amounts of data from a local server using a direct or mobile SOCKSv5 proxy server.

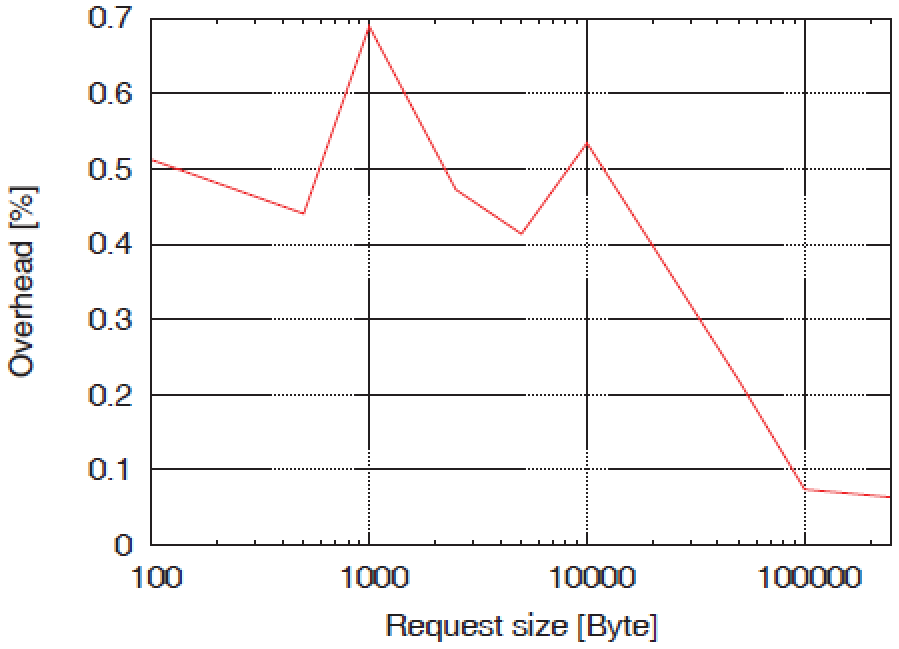


Figure 7. Average energy consumption overhead for requesting different amounts of data from a local server

noted that both depicted connection methods follow a âĂIJslumpâĂİ-like behavior. Taking the overall variability of measurements into account, until the amount of requested data approaches the single packet payload region, the lowest energy usage is recorded. This initial behavior can be explained by the overhead to establish the initial server connection, which is dominant for the small request size regions. With a further increase of the payload to very large sizes, the number of required transmissions drastically increases, which now accounts for the majority of the energy consumption (rather than the initial connection setup). This increase in the number of packets is now causing longer periods of energy expensive network activities, causing the rise in energy consumption illustrated. The overall absolute difference between the employment of a mobile proxy and direct connection, however, remains fairly constant, which indicates that the processing overhead for local packet processing prior delivery to an application is rather negligible. Next, the overhead incurred by the mobile proxy server (as in Equation 1) in Figure 7.Initially, an overall âĂIJhumpâĂİ-like behavior is observable which decreases as the request sizes increase. Part of this behavior is explained by the rather large variability (which isn't illustrated here for clarity) in the regions of smaller request sizes. The overall diminishing overhead is explainable by the increase in the web request sizes with the actual overhead in the processing and local loop-back data connections for the proxy service. As the proxy service requires an initial connection setup before performing the pass-through connection from the remote server, additional time and processing resources are required before the connection can be fully passed through the proxy. Both resources incur processing and delay penalties which in turn have negative power consumption impacts. Larger request sizes result in increased numbers of packets to transmit, which âĂIJsmoothesâĂİ the fixed setup overhead over more packets, which results in the decreasing overhead. At 250 kB of requested data, an observable reduction occurs to just below six percent. Overall, that based on these measurements, it's observed that the implementation of a mobile proxy server directly on a mobile device does not result in considerable additional power demands for larger-sized web requests, which are common today.

Delay Difference

With most interactive settings, the overhead in terms of additional delays can have a negative impact on user utility or perceived quality of service. In networked mobile settings, increased delays additionally have a negative battery impact through the direct correlation between the two. The average delay differences between the proxy and the direct connections for different request sizes is illustrated in Figure 8. Initially, there is a fairly steady level of added average delays to the mobile proxy service in the region of 25 ms with higher levels of standard deviation occurring where the larger amounts of data requested; an overall maximum of about 37 ms can be noted for requesting 250 kB of data. The variability for the delays is a combination of the script producing the larger chunks of data with now slightly increased variability, as well as the increased number of packets sent for each approach incurring an additional network delay variation. Overall, it can be concluded that the observed low level of additional delay represents the fairly constant overhead of setting up the proxy service on the mobile device and processing the initial connection requests.

Media Streaming

The next step is to evaluate the impact of a long duration connection through the mobile SOCKSv5 server, in contrast to the previous evaluation of frequent connection requests. Here, a continuous data stream needs to be forwarded to a local application, with the initial setup overhead becoming negligible. The playback of the web video on the Pandaboard is performed using the Firefox for Android web browser. For measurements of the incurred proxy overhead, the browser is reconfig-

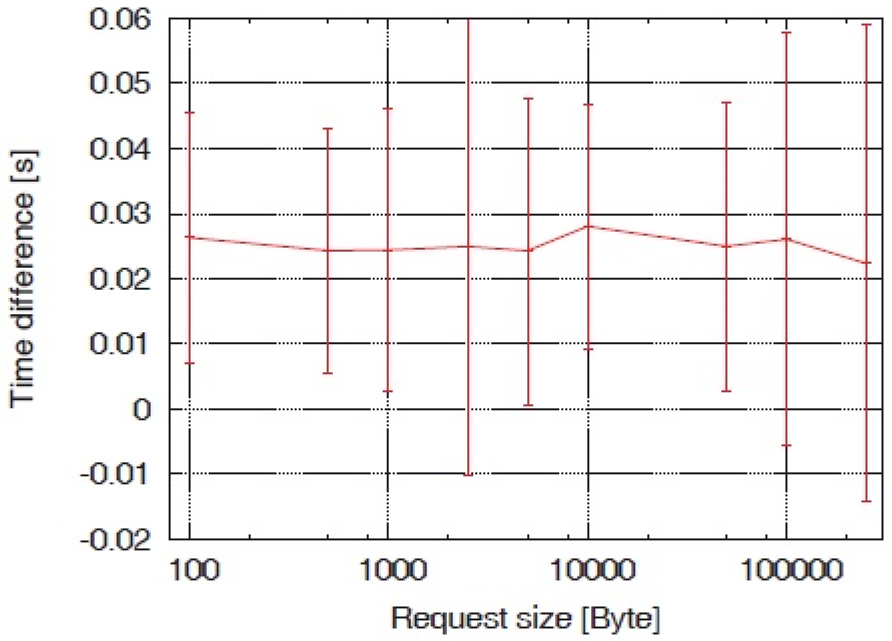


Figure 8. Average time difference and standard deviation between the direct and the proxy approach for requesting different amounts of data from a local server.

ured to utilize the local SOCKSv5 proxy service, similar to the previous web request scenarios. The HTML5 video is in turn displayed on the browser for the entire duration.

Fixed Network

The sampled power consumption for video streaming over a wired Ethernet network is illustrated in Figure 9. Observing the graph, we can see that most of the measured power consumption values fall into a range between 3.25 W and 3.75 W for both approaches. Both approaches additionally exhibit overall periods of heightened power consumption, e.g., from about 250s to about 290s. This time frame coincides with a fast camera movement following an actor (very high level of background content change, motion) and represents a more challenging video decoding task, in turn leading to higher power consumption. Comparatively, however, it’s also observable that no significant differences exist between the two approaches. The aggregated energy consumption resulting from media streaming results in a linear behavior, which additionally is almost indistinguishable between the two

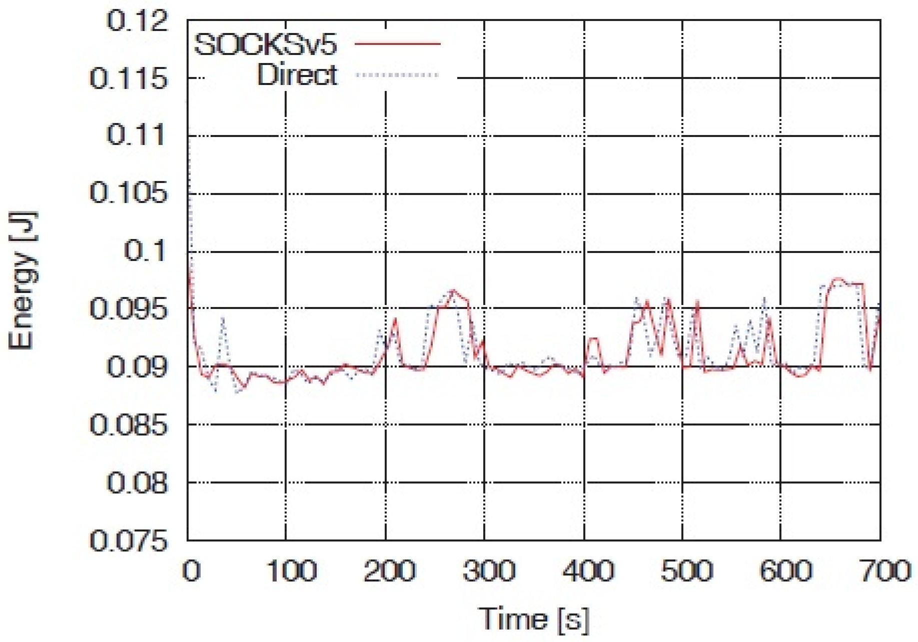
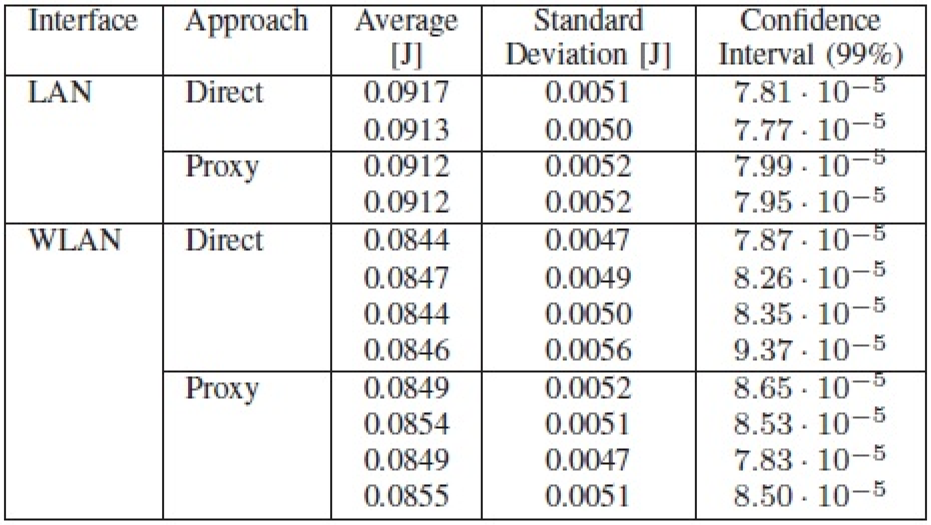


Figure 9. Energy consumption while performing the video streaming of the Tears of Steel open-source video sequence directly and through a mobile SOCKSv5 proxy server, smoothed over time.

approaches as observed for the prior web requests scenario. As indicated by the comparable energy consumption over time in Figure 9, the additional video decoding power requirements let the overall power consumption differences remain minor. We in turn omit a visual representation that closely resembles Figure 5 due to space constraints. The aggregated performance values for this scenario in Table 10. NEED TO CHANGE THIS TO TABLE FORMAT IN LATEX INSTEAD OF PICTURE) Looking at the table, I note that the average energy used is very similar for either approach, with slightly elevated values for the direct approach. Taking the standard deviation and the very narrow 99% confidence intervals into account, we note that the proxy-based streaming incurs no significant penalties. Comparing this streaming example to the individual web requests, I note that overall, the proxy-based approach now is at the same overall level that I observed for the direct case.



Interface	Approach	Average [J]	Standard Deviation [J]	Confidence Interval (99%)
LAN	Direct	0.0917	0.0051	$7.81 \cdot 10^{-5}$
		0.0913	0.0050	$7.77 \cdot 10^{-5}$
	Proxy	0.0912	0.0052	$7.99 \cdot 10^{-5}$
		0.0912	0.0052	$7.95 \cdot 10^{-5}$
WLAN	Direct	0.0844	0.0047	$7.87 \cdot 10^{-5}$
		0.0847	0.0049	$8.26 \cdot 10^{-5}$
		0.0844	0.0050	$8.35 \cdot 10^{-5}$
		0.0846	0.0056	$9.37 \cdot 10^{-5}$
	Proxy	0.0849	0.0052	$8.65 \cdot 10^{-5}$
		0.0854	0.0051	$8.53 \cdot 10^{-5}$
		0.0849	0.0047	$7.83 \cdot 10^{-5}$
		0.0855	0.0051	$8.50 \cdot 10^{-5}$

Figure 10. Summary values for HTML5 video streaming of the open-source Tears of Steel movie (video-only) over an ethernet and a wireless network.

Wireless Network

As in the previous web request scenario, I additionally investigate the alternative approach whereby the video streaming is now utilizing a local wireless network. The measurement results in Table 10 for equal numbers of measurement points. (Also note that for means of better comparison, the number of measurement points are limited to be equal.) I note that the overall averages are fairly close and furthermore characterized by relatively narrow confidence intervals. The resulting overhead, when comparing the averages of the direct and proxy-based video streaming is 0.0081, or less than one percent. Comparing the overall WLAN levels to the wired measurements, I note a decrease in the overall power consumption. This behavior is explained with the full availability of all network interfaces, whereby the employed operating system seems unable to properly put all different interfaces into a nopower state. Overall, I conclude that the introduction of the mobile proxy server has only limited negative impacts on the power consumption of about one percent when streaming multimedia due to the significantly higher impact of the decoding processes on the energy consumption and the negligible packet processing overheads.

Conclusions and Future Work

I approximated the energy consumption overhead of mobile optimization frameworks through a mobile SOCKSv5 proxy server as a low-end baseline. The selected implementation is based on JAVA and performs all network traffic forwarding on the application layer within the Dalvik VM for Android devices. I found that the overall usage of a proxy service on the mobile device in a web request scenario does not incur any significant power consumption penalties. For HTML5 video streaming (continuous packet processing), I note an overhead of about one percent. I determined that any existing overhead is relatively constant and explained through the initial communication setup overheads, rather than ondevice processing overheads, which were found to be approximately one percent. A basic optimization framework would in turn only need to overcome this very low overhead to enable energy savings potentials.In future research avenues, I plan to investigate further energy savings potentials based on mobile device proxying, in combination with caching and application layer content awareness in tandem with a remote server.

CHAPTER IV

Web Cache Object Forwarding From Desktop to Mobile for Energy Consumption Optimizations

Introduction

With the beginning of the 21st century, networking support for wirelessly connected mobile user devices has fueled a continuous increase in the demand for mobile data. Web requests now originate in a majority from wirelessly connected user devices – a trend that Cisco, Inc. predicts to continue in the foreseeable future [1]. Simultaneously, the overall user behavior and demand for more rich media inclusion into web pages has increased the overall amount of data that is required to be transmitted per page, see, e.g., [10, 11]. Caching on the client side has been effectively used in the past and was, together with increased numbers of parallel object downloads, able to decrease wait times for desktop clients as reported in [10]. A first view of mobile web page characteristics (which were found to exhibit lower complexity than regular desktop browser versions) and non-landing pages (which were found to be less complex than landing pages) was evaluated by the authors of [11]. Moreover, a significant body of research has emerged that focuses on content delivery optimizations to mobile devices.

Typically, these optimization approaches are targeting on-device optimizations or off–device cloud–based improvements. For mobile applications, for example, significant energy savings were found to be attainable when grouping application requests so as to avoid prolonged cellular network interface activity, see, e.g., [12, 13]. Other approaches optimize the delivery to mobile devices through proxies and cloud-based data anticipation and traffic shaping, see, e.g., [14].

For web data, typically caching is used to limit the amount of data that has to be transfered to requesting clients. In prior works focusing on mobile web page delivery optimizations, such as [15],

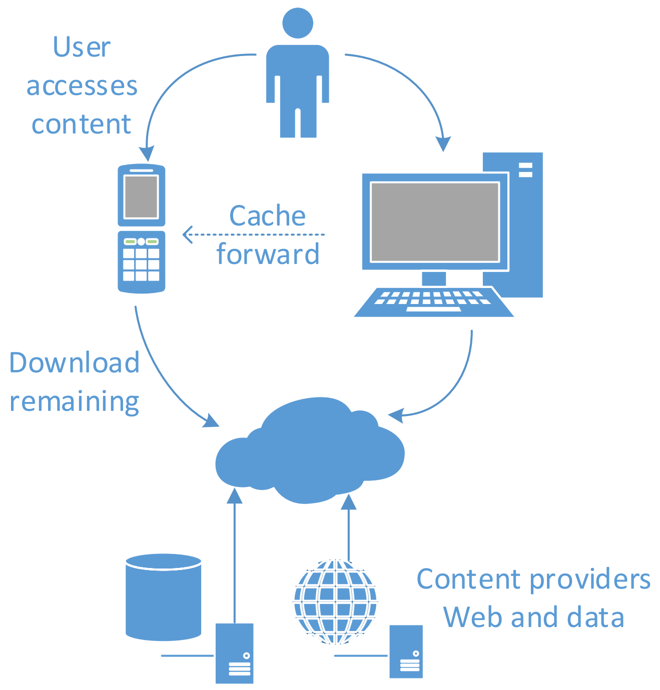


Figure 11. Screenshot of rendered web page from WebPageSpeedTest.org. As illustrated, the main textual and pictorial content items are flanked by background and interactive advertisements.

energy optimization has been a key element, due to the limiting restrictions for mobile clients. One particular area for optimization is the pre–fetching of web page objects combined with caching (which allows to download data before usage), such as presented in [16] with about 10 % energy savings. More recently, these approaches were combined with user connectivity predictions, see, e.g., [17].

We propose to utilize a basic cache forwarding method that can be utilized by users to synchronize from, e.g., a desktop computer, to their mobile device, e.g., a smartphone. We illustrate this approach in Figure 11, which contains the desktop and a mobile device. As most devices are charged over night, or are at least stationary within local area network ranges, we propose to utilize this general idle period to allow direct forwarding of cached objects. We presume that browser clients will have a shared information source in the cloud that allows the identification of visited web pages, an approach most browser clients take to date. In turn, clients are enabled to identify web page objects that are identical between the different device display modalities, i.e., identical web objects delivered when requesting a desktop/mobile versions of a web page. The thus transmitted objects, in turn, reside locally within the mobile device cache and do not require an energy–expensive mobile download through cellular interfaces. We note that the reverse of this approach is possible as well. To support our approach, we gather data through the publicly accessible WebpageSpeedTest.org website as well as the httparchive.org archive of a large dataset of performance evaluations for popular web pages; we refer the interested reader to [**?**] for a more detailed discussion.

CHAPTER V

CONCLUSION

Future Work

REFERENCES

[1] Cisco, Inc. Cisco visual networking index: Global mobile data traffic forecast update, 2013–2018. Technical report, February 2014.

[2] Feng Qian, Zhaoguang Wang, Yudong Gao, Junxian Huang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 51–60, New York, NY, USA, 2012. ACM.

[3] Andre Luiz Tinassi DAmato, Linnyer Beatrys Ruiz, Anderson Faustino da Silva, and Jose Camargo da Costa. Eprof: An accurate energy consumption estimation tool. In *Proceedings of the 2011 30th International Conference of the Chilean Computer Science Society*, SCCC '11, pages 210–218, Washington, DC, USA, 2011. IEEE Computer Society.

[4] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIX-ATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[5] N. Davies, M. Langheinrich, R. Jose, and A. Schmidt. Open display networks: A communications medium for the 21st century. *Computer*, 45(5):58–64, May 2012.

[6] Patrick Seeling. An overview of energy savings potentials through mobile forwarding proxy framework. *Int. J. Ad Hoc Ubiquitous Comput.*, 16(4):260–267, September 2014.

[7] Patrick Seeling. Caching proxying for mobile users. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, pages 1–5, June 2013.

[8] "M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones". Socks protocol version 5. RFC 1928 (Proposed Standard), Mar 1996.

[9] K. Kouzoubov. Java socks proxy. Available: http://sourceforge.net/projects/jsocks/, 2011.

[10] Sunghwan Ihm and Vivek S. Pai. Towards understanding modern web traffic. In *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 295–312, Berlin, Germany, November 2011.

[11] M. Butkiewicz, H.V. Madhyastha, and V. Sekar. Characterizing web page complexity and its impact. *Networking, IEEE/ACM Transactions on*, 22(3):943–956, June 2013.

[12] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications.

In *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 280–293, Chicago, IL, USA, November 2009.

[13] Feng Qian, Zhaoguang Wang, Yudong Gao, Junxian Huang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *Proc. of the International Conference on World Wide Web (WWW)*, pages 51–60, Lyon, France, April 2012.

[14] Yu Xiao, Pan Hui, Petri Savolainen, and Antti Ylä-Jääski. Cascap: Cloud-assisted context-aware power management for mobile devices. In *Proc. of the International Workshop on Mobile Cloud Computing and Services (MCS)*, pages 13–18, Bethesda, MD, USA, June 2011.

[15] F. Sailhan and V. Issarny. Energy–aware web caching for mobile terminals. In *Proc. of the International Conference on Distributed Computing Systems (ICDCSW)*, pages 820–825, Vienna, Austria, July 2002.

[16] Huaping Shen, Mohan Kumar, Sajal K. Das, and Zhijun Wang. Energy–efficient data caching and prefetching for mobile devices based on utility. *Mobile Networks and Applications*, 10(4):475–486, August 2005.

[17] B.N. Thyamagondlu, V.W. Chu, and R.K. Wong. A bandwidth–conscious caching scheme for mobile devices. In *Proc. of the IEEE International Congress on Big Data (BigData Congress)*, pages 78–85, Santa Clara, CA, USA, June 2013.