Anatomizing Application Performance Differences on Smartphones

Junxian Huang[§] Qiang Xu[§] Z. Morley Mao[§] Ming Zhang[†] Paramvir Bahl[†]
[§] *University of Michigan* [†] *Microsoft Research*

Abstract – The widespread deployment of 3G technologies and the rapid adoption of new smartphone devices like *iPhone* and *Blackberry* are making cellular data networks increasingly popular. In addition to email and Web browsing, a variety of different network applications are now available, making smartphones potentially reasonable substitute for their desktop counterparts. Unfortunately, the performance of smartphone applications, from the perspective of the users and application developers, is still not well understood.

We believe our study, the first of its kind, fills this void. We identify and study important factors that impact user perceived performance. We formalize the method for comparing the performance of smartphone applications along several unique dimensions such as carrier network, device capabilities, application types, and network protocols. To ensure a fair comparison across platforms and networks we develop a detailed measurement methodology. Our work is an important and necessary step towards understanding the performance of smartphone applications from users and application developers perspective. Our analysis culminates with a set of recommendations that can lead to better application design and infrastructure support for smartphone users.

1 Introduction

As of the third quarter of 2008, global smartphone shipments reached 40 million units representing 13% of the total mobile phone market [2]. It is widely accepted that in the next few years smartphones sales will outpace the sales of regular phones. Vendors such as Research in Motion, Samsung, Nokia, HTC, and Apple offer a variety of smartphones that are equipped with fast CPU and large memory. These phones support various high-speed 3G networks such as EVDO and UMTS. They are powerful enough to run modern operating systems and sophisticated network applications such as Web browsing, email, and streaming media.

The performance of network applications on smartphone depends on several factors including the quality of the hardware, the software, the wireless link, and network traffic load. Understanding the performance of applications on smartphones is an important and challenging problem. This understanding can assist consumers in selecting the carriers and phones on which their favorite applications perform well and can guide application developers to write smarter software. Also, cellular network operators, smartphone hardware and software vendors can use this knowledge to optimize their networks and phones in a way that lead to better end-user experiences.

Researchers have done extensive work in measuring and optimizing TCP performance over cellular links. For example, Liu *et al*. [7] studied multiple TCP variants by correlating the link-layer measures, *e.g.,* SINR (signal-to-noise ratio) and DRC (data rate cover), with TCP performance. While many applications are built on TCP, their actual user-level performance does not match that of TCP alone due to various adaptation strategies and overheads induced by the applications. Realizing this problem, Chesterfield *et al*. [4] evaluated the performance of streaming media application over a WWAN (Wireless Wide Area Network). They studied how link-layer characteristics impact the inter-packet arrival time, bandwidth, and buffer delay of streaming media. But their work is limited to a customized streaming application named *vorbistreamer*. In a separate study, Chakravorty *et al*. [3] measured the performance of TCP and Web browsing over WWANs. However, that study is also different from ours because they focus only on comparing the overall throughput across different link layers.

We focus on developing a systematic set of methodologies and tools for measuring and analyzing the performance of smartphone applications in a way that is directly relevant to end users. As a consequence, (1) we measure the performance of applications instead of just low-level protocols. Prior work has shown that application performance often significantly deviates from protocol performance [14]. We target the pervasive web browsing application that most end-users care about; (2) We measure the application performance on several different mobile devices that consumers actually use. We find that due to the differences in hardware and the platform software, application performance varies widely across different devices. This then leads us to study and compare the performance of application both on laptops (as has been done in most prior work) and on smartphones; (3) We study the application performance under real-world scenarios. We quantify the performance of Web browsing by measuring the performance of accessing real Web services instead of just the ones under our control (once again, this differs from prior work). We do perform some experiments using our own web site but those are only for the purpose of dissecting the performance implications of the dynamic interactions between consumers and real Web services.

In addition to shedding light on the overall application performance, we perform detailed analysis to help carriers, hardware vendors, operating system vendors, and

application developers gain insight on the factors that impact user-perceived performance. For carriers, we infer various network-level problems, *e.g.,* high latency or high loss rate, which they can directly take action on. For hardware and software vendors, we identify issues with the devices or the customized contents. They can resolve these issues either independently or by cooperating with content providers, and for application developers we identify issues such as the impact of HTML parsing and Javascript executions.

We comprehensively study the UDP, TCP, and Web browsing performance for four major US carriers including AT&T, Sprint, Verizon, and T-Mobile. These four carries operate both 2G and 3G networks. For devices, we use iPhone from Apple Inc., Windows Mobile phones from Palm, HTC, and Samsung, and desktop computers for carrying out our experiments. Our results show that their performance varies significantly across different Web services. In fact, even for the same Web service, the desktops and certain types of phone consistently outperform others due to the difference in downloading behavior, customized contents, and page rendering techniques. We show that the performance also heavily depends on various factors in carriers' network including DNS lookup, RTT (round trip time), and loss rate.

Below, we summarize our main observations obtained after extensive experimentation. Some of these are consistent with expectations, while some are non-intuitive and even surprising.

1. The same set of smartphones accessing the same set of websites exhibits performance differences among one other. We attribute this observation to several factors including customization of content based on device type, web servers transferring web objects in different compression modes, and differences in browser concurrency, page rendering delay at the client, object download speed, DNS lookup delay, and TCP three-way handshake delay.
2. For the same cellular data network, device (and operating system) specific differentiation appears to exist, even after we compensate for end-system software issues.
3. Network performance differs between laptops that use the phone as a modem and phones that access the Internet directly. We attribute these differences to the differences in hardware capability and software features between the two set-ups. Previous studies have ignored this fact and presented results for the former case only.
4. All the smartphones can reap significant performance benefit from WiFi Ad-Hoc proxy based Internet access, However, this cannot eliminate application performance differences due to many non-networking issues, such as page rendering delay at the client, browser concurrency, and content customization.

In the rest of the paper we elaborate on these observations, share additional ones and discuss the implication on application design, network operation, and platform design.

2 Related Work

We were influenced by the NetDiff system [8], which established a benchmark for comparing performance of different Internet Service Providers (ISPs). In our research we attempt to establish an equivalent benchmark for 3G carriers based on the network application performance observed on the smartphone and laptops. Although some user-based online comparisons are available [1], we believe that ours is the first comprehensive study that analyzes and compares the network behavior of both 2G and 3G networks from the perspective of the applications executing on end-user devices.

There are several studies that have examined cellular data networks but their focus was different. Examples include, a study of the interaction between the wireless channels and applications [7]; of application-aware acceleration to improve application performance [14], and of performance of multimedia streaming [4]. Our work is the first to evaluate 3G cellular network performance directly using phones as the platform for measurement, thus it accurately reflects the actual user experience. All previous studies (*e.g.,* [7, 3]) perform measurements on the desktop or laptop systems, relying on cellular network data cards or phones connected through a USB as a modem. We show later in the paper that the performance measured on laptops versus that measured on phones for the same network is different. Phones have more severe resource restrictions, execute potentially different types of applications, and send and receive different application content compared to their laptop and desktop counterparts.

Unlike many previous works, we mostly take a black-box approach to performance measurement by examining network behavior at the application, transport, and network layers without relying on detailed information of the wireless channels (*e.g.,* [7]) nor the internal state of cellular data networks (*e.g.,* [13]). This then presents an interesting challenge of inferring the bottleneck or the root cause for observed application performance. We argue that this approach will stand the test of time as it is better suited for periodic analysis of evolving networks. The limitation it has, of reduced visibility into the inner working of the network, does not prevent us from achieving the goal of effectively comparing network performance across different cellular data network providers.

Our work builds on numerous previous TCP studies for cellular data networks which aim to understand the behavior of TCP using cross-layer measurement techniques [9], modeling of multi-rate and multi-user behavior [6], and potential transport improvements for wireless wide-area networks [12]. These studies expose the limitations of existing designs and our work confirms some of these problems as measured from the perspective of end-users.

Finally, previous work has also proposed optimizations at multiple layers to improve the performance of streaming media [4], Web browsing [3, 11], and a few other mobile wireless applications [14]. Wherever applicable, we analyze and discuss how effective these optimizations are to the extent that they are visible in our experiments.

3 Experimental Methodology

In this section, we present our methodologies for measuring network and application performance over 3G wireless networks. To help end-users make informed choices regarding which carrier to use, we measure the overall performance of popular web applications. The overall application performance may depend on many factors across layers, *e.g.,* RTT, loss rate, DNS lookup, browser concurrency, content organization, and client execution. To help carriers and application developers identify performance bottlenecks and make targeted improvements, we further perform detailed analysis on the measurement results.

3.1 Measuring network performance

Measuring the throughput of TCP and UDP over 3G links can provide valuable insights to the network operators about the channel capacity available to users. Application developers also care about these two measures because many applications, *e.g.,* video streaming and file transfer, are built on top of TCP or UDP.

To measure UDP throughput, we use constant bit rate (CBR) flows because CBR is not impacted by network condition variations. In the ideal case when the sending rate matches the channel capacity, the available bandwidth will be fully utilized. In our experiments, we vary the sending rate incrementally and use the maximum throughput measured at the receiver to estimate channel capacity.

To measure TCP throughput, we use a long-lived TCP flow. The measured TCP throughput may be lower than the available channel capacity because of TCP congestion control. The actual TCP throughput also depends on TCP implementations, *e.g.,* how congestion window is adjusted over time or how packets are retransmited during loss events. Besides throughput, we extract the loss rate and RTT for each TCP flow. These two metrics help to explain why throughput is high or low. We can infer RTT and loss rate from packet traces.

3.2 Measuring web application performance

One of the most widely-used applications among smartphone users are web based applications. There are numerous traditional web portals as well as web-based services, *e.g.,* search, email, and map. Almost every type of smartphones includes web browsing allowing us to compare performance across different phones.

When a user visits a webpage, the browser first performs a DNS lookup to obtain the IP address of the web server. It then establishes a TCP connection with the server before it starts to download the main webpage. The main page may embed many web objects, including CSS, Javascripts, and images, which sometimes are hosted by servers in multiple domains. In that case, the browser has to perform more DNS lookups, establish multiple connections to different servers, and download objects in parallel. This process continues recursively until all the objects are downloaded. Clearly, the page load time depends on factors such as DNS lookup, TCP handshake, TCP transfer, and client execution.

The contents of many webpages are quite dynamic. The structure and embeded objects in a page may evolve over time, reflecting the addition of new contents or changes in page design. Even within a very short period, *e.g.,* a few seconds, there could be minor changes when a page is loaded multiple times. This is often due to certain dynamic web objects that are generated on-demand, such as advertisements. Some pages even provide customized contents based on the types of phones in order to optimize user experience. Such content variations can be problematic for comparing performance across different phones and across different times.

In our experiments, we measure the performance of loading a webpage multiple times to alleviate the impact of random noise. We make sure the measurements of the same page are completed in a short time period, *e.g.,* a few minutes. Furthermore, all the phones load the same page around the same time. These two steps help to reduce the chance of being affected by significant content variations in a page. After the measurements are completed, we further verify that the downloaded contents of each page are similar across different phones and across different runs. In case a page provides customized contents for a particular type of phone, we force that phone to download both the customized and the regular contents in order to evaluate the impact of content customization on performance.

To perform detailed analysis of web application performance, we extract the following information from the packet trace of a page download:

Page load time is the time between the first DNS packet and the last FIN packet from the server during a page download. It reflects the overall performance perceived by a user. Note that browser needs to further parse and render a webpage after it is downloaded. The parsing and rendering time may not be included in page load time. In § 7, we introduce a method to measure the parsing time of an HTML page.

Page size is the total number of unique downloaded bytes. It can be used to compute *average throughput* and to detect content variations and customizations.

Connection completion ratio denotes the percentage of complete TCP connections. A TCP connection starts with an SYN and ends with a FIN, a RST, or a timeout. We consider a connection to be complete if it ends with a FIN. We discard a page download sample if its complete connection ratio is low ($< 95\%$).

Loss rate & RTT are extracted from the trace for each TCP connection. We aggregate the loss rate (denoted as p) and the RTT of individual connections to

produce an overall measure of network performance of a page download. Given that TCP throughput can be modeled as $\frac{MSS}{RTT\times\sqrt{p}}$ where MSS is the maximum segment size [10], the download time of a connection of size S can be estimated as $\frac{S\times RTT\times\sqrt{p}}{MSS}$. Since the download time of a connection is proportional to S, RTT, and $\sqrt{p}$, we use $\frac{\sum(S_i\times RTT_i)}{\sum S_i}$ and $(\frac{\sum(S_i\times\sqrt{p_i})}{\sum S_i})^2$ as the average RTT and loss rate of a page download.

Browser concurrency Most browsers support concurrent TCP connections within the same domain to improve download efficiency. The maximum number of concurrent TCP connections within a domain varies by browsers. Because each connection has its own start and end time, we compute the average concurrency of a page load as the total duration of all the connections divided by the page load time. When the network is not the bottleneck, higher concurrency usually means better utilization of bandwidth which in turn leads to shorter download time.

DNS lookup time A page load may involve many DNS lookups of different domains. We compute the total DNS lookup time to quantify the overall impact of DNS lookup on performance. Since DNS lookup requests are handled by local DNS (LDNS) servers, we also compute the average DNS lookup time as a measure of LDNS server performance.

TCP handshake time Each TCP connection starts with a three-way handshake during which no data is transferred. We compute the total handshake time as a measure of overhead induced by handshake.

TCP idle time & transfer time Given a TCP connection, an *idle period* is defined as a period of at least 1 second during which no data is transferred and no retransmission is detected. The remaining periods in the connection are the *transfer periods*. An idle period usually corresponds to the processing time on the phone between the instance when a response is received and the instance when the next request is issued. The processing time is particularly pronounced on the phone due to its limited CPU power and memory. We choose 1 second threshold because it is almost always bigger than the measured RTTs (Figure 5). When TCP has data to send and is not in retransmission, there should be data transmission in every RTT.

There are a few other factors that may affect web application performance, *e.g.,* object compression. In § 7, we will conduct controlled experiments to study the effects of compression on page load time.

4 Experimental Setup

Table 1 lists the devices and carriers used in this work. We study the four major carriers in the US, including AT&T, Sprint, Verizon, and T-Mobile. Among them, T-Mobile only provides 2G EDGE service in our locations. The other three 3G carriers are split between HSDPA/UMTS (AT&T) and EVDO (Sprint and Verizon). AT&T has the highest advertised downlink and uplink data rates, up to 1.7 and 1.2 Mbps respectively. The advertised data rates of Verizon and Sprint are somewhat lower. We cannot find any official advertised data rate from T-Mobile. The actual data rates that a user can attain depend on many other factors, such as signal strength, location, and background traffic. One of our goals is to understand the extent to which the actual data rates match the advertised ones and how such mis-match impacts the relative performance of applications such as web browsing.

Carrier	AT&T	Sprint	Verizon	T-Mobile
Network	UMTS	EVDO	EVDO	EDGE
D(Mbps)	0.7-1.7	0.6-1.4	0.6-1.4	n/a
U(Mbps)	0.5-1.2	0.35-0.5	0.5-0.8	n/a
Vendor	Apple	Palm	Samsung	HTC
Device	iPhone	Treo800w	SCHi760	TyTNII
Mem	128MB	128MB	64MB	128MB
ARM	1176	1136	920T	1136EJS
MHz	620	333	400	400
OS	OS 2.1	WM 6.1	WM 6.1	WM 6.1
Browser	Safari	IE	IE	IE

Table 1. Carriers and devices

§	Experiment	Description	#
5.1	UDP performance	Transport layer	12
5.2	TCP performance		
6.1	Video streaming	App overall	6
6.2	Web browsing		
7	DNS lookup	App breakdown	7
	TCP handshake		
	Object download		
	Javascript execution		
	HTML parsing		
7	Browser concurrency	Controlled scenario	5
	Object compression		
	WiFi proxy		

Table 2. Summary of experiments

To measure user-perceived performance on devices, we conduct our experiments on four popular smartphones listed in Table 1. The three Windows Mobile phones and iPhone are different in terms of CPU, memory, OS, and browser. The hardware and software differences between the phones are among the most dominant factors that contribute to the application performance differences. In the case of web browsing, the CPU speed and OS scheduling algorithm determine how fast Javascripts or HTML objects are executed. Browser concurrency influences how efficiently the available wireless channel capacity is utilized. Different phones may also receive customized contents that directly affect both execution time and data transfer time.

Beside the four phones, we also use 2 desktops in three different ways in our experiments: i) a client that uses phone as a modem; ii) a server that resides on the Internet; and iii) a proxy that provides high-speed Internet connection to a phone via WiFi. The desktops have Intel Core2 Duo 2.26GHz processor and 2GB memory. They run Windows Vista and IE 7 browser.

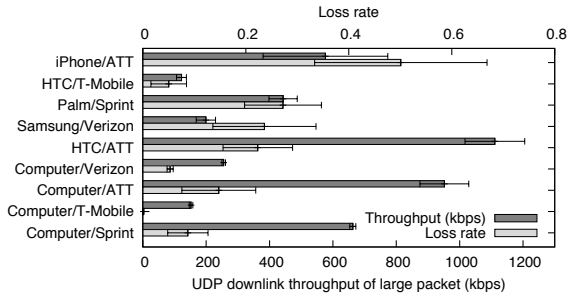


Fig. 1. UDP results across platforms and carriers .

Table 2 summarizes the experiments and the corresponding sections in this paper. We measure both network performance and application performance in order to understand the correlation between them, *e.g.,* whether a phone with higher TCP throughput will also have smaller page load time. Because application performance does not always correlate well with network performance, we conduct controlled experiments to identify other significant factors, such as DNS performance, TCP handshake performance, and page rendering performance. Moreover, we study the impact of browser concurrency and the tradeoff between network and CPU overheads of object compression. Finally, we perform a case study on to what extent using a WiFi proxy can help to improve user experience compared to using 3G network.

We implement a suite of tools to measure the performance of TCP, UDP, DNS, and object download on four phones. They are written in Java on iPhone and C++ in Windows Mobile. To capture packet traces for detailed analysis, we use *tcpdump* on iPhone and *Netlog* on Windows Mobile. The format of Netlog traces is the same as that of tcpdump.

Because a 3G link can be quite unstable, we repeat each experiment multiple times to discount the impact of random noise (column 4 in Table 2). We report both mean and standard deviation for results that have significant variations. We repeat some of these experiments at different times of day and different locations to verify the consistency of our observations. To avoid potential interferences by mobility and power management, all the phones are kept stationary in the same location with plugged-in power during the experiments.

5 Network Performance

To understand the different characteristics of network applications, we first focus on controlled network experiments to test the throughput of TCP and UDP based data streams. We infer the bottleneck resources by conducting Iperf-like one-way data transfer for exposing sustainable network throughput between the phones and a well-provisioned host in the wired network. Each throughput test lasts for 5 minutes. We measure UDP throughput with packet size of 1300 bytes. We correlate RTT and loss rate to explain the observed throughput of TCP data streams. And we also compare across devices and networks to identify and explain observed differences in performance.

5.1 UDP performance

Unlike TCP, UDP does not guarantee reliable and in-order delivery, and is more commonly used for multimedia streaming, *e.g.,* YouTube. We evaluate the maximum throughput sustained for UDP streams for downlink network behavior. Since the desktop machine communicating with the phone for performing the experiment is connected to 100Mbit Ethernet wired network, we expect the bottleneck link in terms of latency and bandwidth to reside inside the 3G networks. Figure 1 shows the downlink throughput behavior collected mostly during night hours at our location.

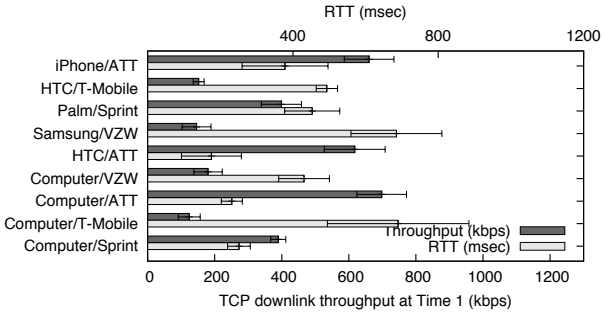
Cross phone, cross carrier: We compare across the five phone platforms with corresponding carriers. The downlink throughput values shown in Figure 1 range from about 150kbps for HTC/T-Mobile to about 1100kbps for HTC/AT&T. T-Mobile's network offers only 2G service at our location; however, the corresponding number for Samsung/Verizon phone is only 200kbps, one fifth of the rate of HTC/AT&T. These maximum UDP throughput are the saturated UDP throughput values observed by increasing the sending rate. The loss rate observed is expected to be very high, *e.g.,* close to 50% for iPhone/ATT with high variability, though the exact values shown may vary greatly depending on the saturation point chosen. Thus, we do not compare them at a fine-grained level.

The clear difference across carriers is also reflected by comparing iPhone/AT&T with HTC/AT&T, where HTC/AT&T consistently outperforms, in fact for TCP and UDP traffic. We do not expect AT&T to intentionally differentiate traffic based on the actual phones, we expect the differences be caused by factors directly related to the phones, *e.g.,* software such as the network stack or hardware such as the radio.

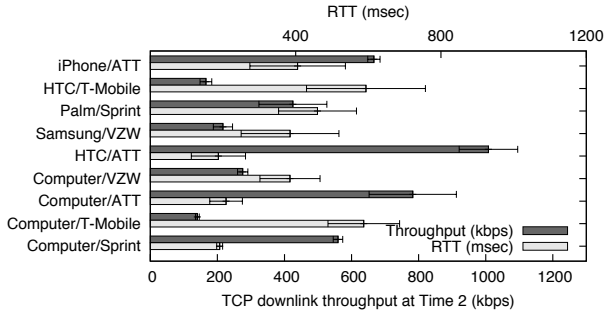
Phone vs. computer: Previous studies have performed measurements using desktop or laptop systems connected to the cellular data networks either via data cards or through the phones acting as a modem. We compare the UDP throughput behavior measured on the computer with that on the phones. Given that the phones are much more restricted by limited resources, we expect the desktop to perform better. This is mostly confirmed by all plots in Figure 1, demonstrating that in some cases, such as Sprint, the computer based performance improved by almost 50% – 662kpbs vs. 442kbps. This confirms our earlier conjecture that performance observed on computers do not truly reflect that experienced by phone users, thus validating the motivation for our study.

5.2 TCP performance

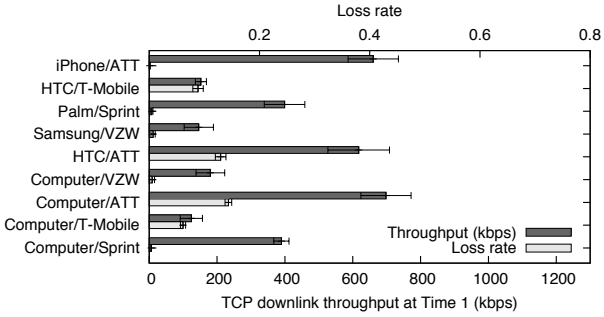
Most applications use TCP as their transport protocol; thus, TCP behavior is critical to the overall performance of network applications. Previous work has proposed various TCP variants optimized for wireless networks. Our study focuses on the behavior of the TCP stack currently running on Windows Mobile phones and iPhone platforms without detailed knowledge of the actual TCP



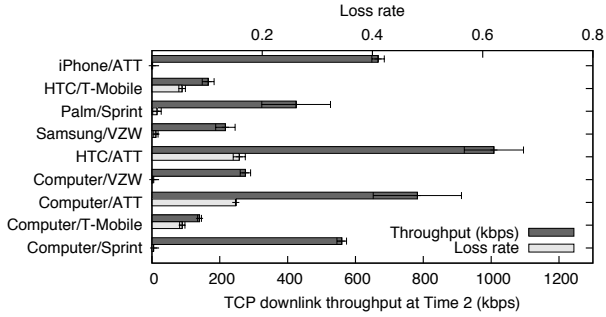
(a) Throughput and RTT correlation (Time 1).



(b) Throughput and RTT correlation (Time 2).



(c) Throughput and loss rate correlation (Time 1).



(d) Throughput and loss rate correlation (Time 2).

Fig. 2. TCP downlink throughput results across two time periods.

implementations. Our data are collected during two time periods: day time (Time 1) and night time (Time 2) to illustrate the variability across times. We examine the downlink TCP throughput characteristics by correlating with RTT and loss rate, as stable TCP throughput can be modeled roughly inversely proportional to the RTT value and to the square root of the packet loss rate value [10].

Figure 2 illustrates the TCP downlink throughput across two time periods for different phone and carrier combinations, as well as for computer based setups. There exist obvious differences in average TCP throughput across the carriers with HTC/T-Mobile clearly lagging behind due to no 3G network access at our location. The overall TCP throughput values for phones range from about 150 kbps to 1.1 Mbps, which is lower than measured downlink rates for broadband DSL or cable networks [5]. The throughput values for Time 2 are generally slightly larger compared to those for Time 1, which is expected due to fewer users expected during Time 2, *i.e.,* the night time. Interestingly, despite its 3G network, Verizon's downlink performance is not much better compared to the T-Mobile's 2G network.

RTT correlation: Large RTT values usually indicate long queuing delays and also result from large buffer sizes. The RTT values observed shown in Figures 2(a) and (b) range from 200 ms to close to 750ms, which are significantly larger compared to typical RTT values on the Internet. Comparing with ping-based probes generating RTT values ranging from 140 to 340 ms depending on the platform, the additional delays are most likely due to queuing inside router buffers. The general trend of larger RTTs matching smaller throughput does hold well in the data we collected.

Loss rate correlation: Figures 2(c) and (d) show that the observed loss rate is much lower than that of UDP flows across all the phones and computer-phone setups with values of at most 15% for HTC/ATT setup. Based on previous studies [7], we conjecture that the low loss rate is a result of aggressive link-layer retransmission and adaptive intelligent coding schemes deployed at the link layer designed to achieve a target error rate, which also uses smart scheduling of channels to avoid collision. TCP adapts to loss experienced as opposed to UDP, further preventing loss at the network layer due to buffer overflows. Also note that the TCP throughput ranking across phones remain relatively the same compared to that for UDP throughput values.

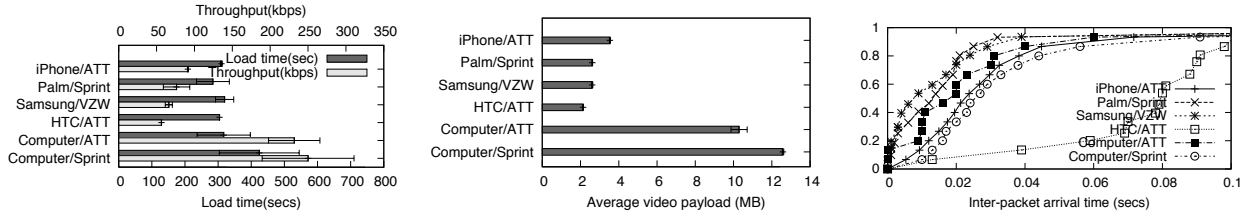
Phone vs. computer: Just as expected and also matching the observations for UDP measurements, the average TCP throughput measured on the phone platform is generally slightly lower than the corresponding measurements on the desktop platform. In some cases, the differences are not too obvious, *e.g.,* Verizon and T-Mobile, due to different resources available on each platform. For Sprint and AT&T, the desktop platform has higher throughput for Time 2.

6 Application Performance

Given the previous discussions on understanding the UDP and TCP throughput behavior, we examine the performance of two common applications on smartphones, namely Web browsing and video streaming. It is important to note that many factors jointly determine user perceived performance, as applications may not fully utilize available network resources and may be limited by pro-

Experiments	Metrics	Rank
UDP uplink	Throughput	HTC/ATT>Sprint>iPhone>VZW>T-Mobile, C≥phone
TCP uplink		
UDP downlink		HTC/ATT>iPhone>Sprint>VZW>T-Mobile, C≥phone
TCP downlink		
YouTube	Load time	Sprint>iPhone>VZW>HTC/ATT>C/ATT>C/Sprint
	Throughput	C/Sprint>C/ATT>iPhone>VZW>Sprint>HTC/ATT
Real web browsing	Page load time	iProxy>C/Sprint≥Sprint≥iPhone>VZW>HTC/ATT>T-Mobile
	Throughput	iProxy>C/Sprint≈iPhone>Sprint>VZW>T-Mobile>HTC/ATT
	DNS	iProxy>Sprint>C/Sprint≥T-Mobile≈VZW≥iPhone
	TCP handshake	iProxy>Sprint>C/Sprint≥VZW>iPhone>HTC/ATT>T-Mobile
	TCP idle time	HTC/ATT>iPhone≥iProxy≥C/Sprint>VZW≥Sprint>T-Mobile
	TCP transfer	iProxy>C/Sprint>VZW>Sprint>iPhone≈HTC/ATT>T-Mobile
	RTT	iProxy>Sprint>C/Sprint>VZW>iPhone>HTC/ATT>T-Mobile
Controlled experiments	Page load time	iProxy>C/Sprint>Sprint≥iPhone≈VZW>T-Mobile>HTC/ATT
	Throughput	iProxy>C/Sprint>iPhone>Sprint>VZW≥T-Mobile>HTC/ATT
	DNS	iProxy≥Sprint≥C/Sprint>VZW>iPhone≥T-Mobile
	TCP transfer	iProxy>C/Sprint>HTC/ATT≥Sprint≥iPhone≥VZW≥T-Mobile
	TCP handshake	iProxy>C/Sprint>Sprint>VZW≥HTC/ATT≥T-Mobile≥iPhone
	Object download	iProxy>C/Sprint>iPhone>VZW>Sprint>HTC/ATT>T-Mobile
	Javascript execution	C/IE7>iPhone/safari=iProxy/Safari>GPhone>Samsung>Palm>HTC
	HTML parsing	C/IE7>iProxy/safari>iPhone/Safari>GPhone>Palm>Samsung>HTC

Table 3. Performance rank order table (iProxy: iPhone-WiFi-Proxy setup, C: computer)



(a) Load time vs. throughput. (b) Total payload differences. (c) Packet interarrival time distribution.

Fig. 3. YouTube video streaming performance across platforms for a 4:57min video clip.

cessing or memory related bottlenecks [14]. Table 3 displays the performance rank order based on the average value among the platforms studied for different settings to be explained in the remaining paper. "≈" means that the average value differs in less than 5%. ">=" indicates that the rank order is not consistent if the distribution of all experiments are considered besides the average.

6.1 Video streaming

We focus on video streaming hosted by YouTube given its popularity. We found that Windows Mobile consistently uses UDP regardless of the phone platform. TCP is used for other platforms including desktop. Figure 3 illustrates the performance of watching a 4:57min video clip, measured using average load and throughput (along with standard deviation), repeated five times for each platform. The load time is computed from the client's SYN packet till the server's FIN packet. As shown, computers clearly outperform phones in throughput but have slightly longer load time due to much larger payload to be downloaded given larger display and higher resolution. Throughput difference may stem from superior processing capabilities on the desktop machines. Figure 3(c) shows HTC/AT&T has surprisingly much larger packet interarrival times explaining its worse throughput. Note that the result for T-mobile is not included, as phones using the 2G network are unable to consistently complete the video clip download.

iPh/ATT	Pm/Spr	Sam/VZW	HTC/TM	HTC/ATT
12	6	5	5	6
C/ATT	C/Spr	C/VZW	C/TM	
4	6	6	6	

Table 4. Max observed concurrency (C: Computer).

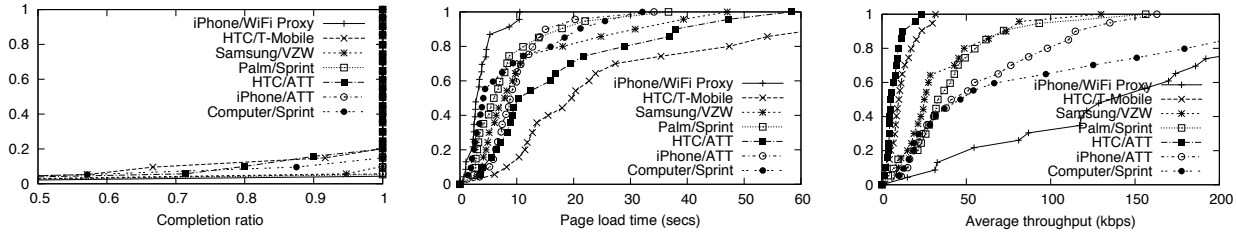
mapquest	hotmail.com	live.com	cnn.com
maps.yahoo	facebook.com	google.com	ebay.com
maps.google	weather.com	amazon.com	espn.com
mail.yahoo	myspace.com	yahoo.com	msn.com
microsoft	youtube.com	gmail.com	nba.com*
wikipedia*	blogger.com	imdb.com*	go.com*

Table 5. 24 URLs used in web browsing experiments

6.2 Web browsing

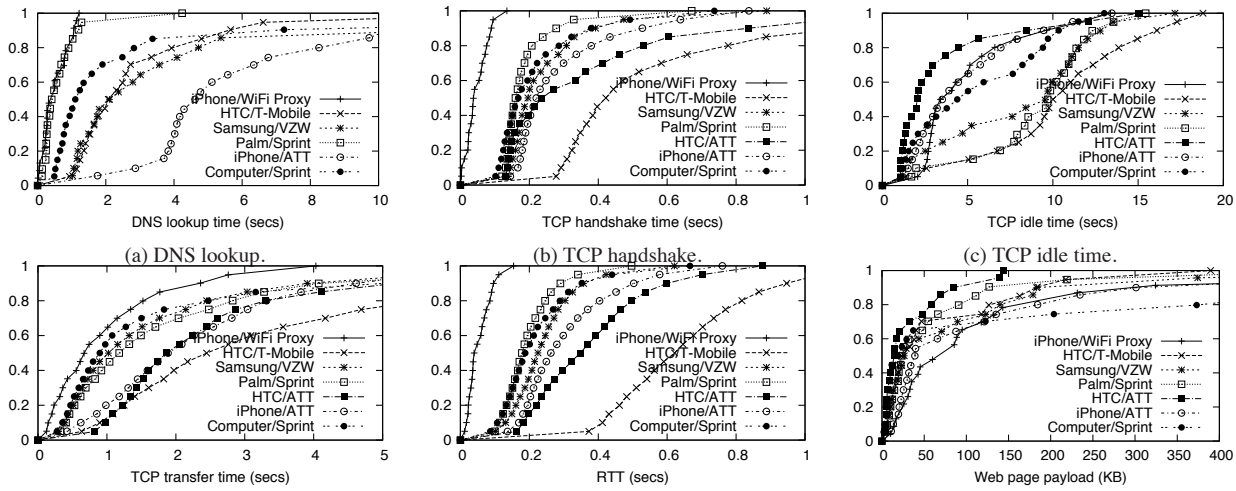
Web browsing is likely the most popular application on smartphones, and its performance is determined by many factors such as network, browser software (*e.g.,* the degree of parallelism), content customization, DNS overhead, CPU resources, *etc*. We attempt to dissect these factors to understand the performance differences across platforms. We briefly describe our experimental setup.

For our study we select 24 URLs from a diverse set of popular content with a variety of content types as shown



(a) Per URL TCP completion ratio. (b) Page load time. (c) Average throughput.

Fig. 4. Overall performance of Web browsing across platforms.



(a) DNS lookup. (b) TCP handshake. (c) TCP idle time.

(d) TCP transfer. (e) Average RTT. (f) Web page payload.

Fig. 5. Breakdown of Web transaction performance: (a)-(d), RTT and Web page payload distribution.

in Table 5, only four of them indicated by '*' do not have customized content for mobile phones. Each is visited 6 times. To facilitate repeated experiments, for Windows Mobile platform, a C++ program is written to invoke IE to visit each URL in turn. We overcome the challenge of the inability to call the Safari browser from iPhone directly by creating a Web page with Javascript that periodically redirects the browser to each URL in turn after 120 seconds.

Figure 4 shows the overall Web browsing performance across the representative platforms studied, omitting some for ease of exposition. To enable fair comparison, we exclude browsing sessions with many incomplete TCP connections, due to reasons such as network or server errors, insufficient wait time, and continuously running Web connection caused by dynamic content. Figure 4(a) shows the distribution per URL TCP connection completion ratio. We use a threshold of 90% to include about 98% of our data samples.

Note that we also show a setup called *iPhone-WiFi-proxy (iProxy)*, in which iPhone connects to a laptop using wireless Ad-Hoc network and the laptop has wired Internet access. iPhone can use the laptop as a proxy to gain Internet access. We compare this setting with the other platforms using 3G networks, and notice significant performance differences.

The page load time distribution shown in Figure 4(b) is quite telling in demonstrating the difference across platforms and the fact that network throughput alone, *i.e.,* measured via TCP downlink Iperf experiments, is not sufficient for predicting application performance. The rank order for Web performance in terms of download time is shown in Table 3. The iPhone-WiFi-Proxy setup is clearly superior for both measures of page load times as well as throughput. HTC/T-mobile has the longest page load time likely due to limited bandwidth in a 2G network compared to 3G networks for other setups. HTC/AT&T again anomalously performs poorly, much worse than iPhone/AT&T. The remaining platforms have similar performance, with iPhone/AT&T slightly ahead of others in average throughput.

It is surprising that although the HTC/AT&T setup has previously shown to have higher TCP downlink throughput than most other platforms (Figure 2), its overall page load time and average throughput lag behind platforms such as iPhone/AT&T and Samsung/Verizon. Upon further inspection we found that HTC/AT&T always contacts a proxy server IP before any URL visits. In our experiments, the IP contacted is `66.209.11.32` with the DNS name of `vdispwap2.mycingular.net`. Such a setup helps improve caching and reduce DNS lookup delay, but also enables ISPs to instrument potential traffic differentiation. We also note that the computer based setups are not consistently better in page download times

for the phone counterparts, likely due to special content customization for the phones.

The average throughput distribution depicted in Figure 4(c) confirms our expectation that iPhone WiFi and computer based setups have higher throughput due to wired network connections. Interestingly iPhone/AT&T appears to have much higher throughput than other setups, despite comparable TCP downlink throughput performance. Our subsequent analysis in decomposing the Web transaction provides further insight for this.

6.2.1 Web transaction breakdown

As previously discussed, performance of Web transactions depends on several factors. To understand the actual bottleneck, we decompose the overall delay in fetching a Web page into the following key components: DNS lookup for resolving domain name in URLs to IP addresses, TCP 3-way handshake for establishing the TCP connection, idle time due to server or client processing, and the actual TCP payload transfer time. These are measured by analyzing network traces collected on the phones or computers. Figure 5 depicts the distribution across platforms for these components. Note that for each of the component below, the values are accumulated across all TCP connections associated visiting a particular URL, then averaged across six visits, clearing the cache after each visit.

DNS lookup: DNS lookup delay is computed by observing packets destined to port 53 with all setups using UDP, except for HTC/AT&T with no such traffic found due to the use of a proxy. There is little concurrent DNS request observed, so we sum up all the DNS request delays for each URL visit. It is interesting to note that iPhone/AT&T actually has the longest DNS lookup times, while iPhone-WiFi-proxy setup and Palm/Sprint appear to have the best performance in DNS. Despite contacting the same DNS server, Computer/Sprint setup has worse DNS delays compared to Palm/Sprint are mainly due to many more lookups for richer content as evidenced by much larger payload in Figure 5(f).

TCP handshake: shows TCP handshake distribution on a per flow basis, matching the rank order shown in Figure 5(e), as TCP handshake is mainly determined by RTT. HTC/AT&T setup has significantly worse average RTT and longer TCP handshake than other platforms, which we suspect is caused by the proxy rather than the 3G network, as explained by better performance of iPhone/AT&T. Again, computer/Sprint does not outperform Palm/Sprint which has slightly larger RTT and TCP handshake values for the top 50 percentile. Again, this is a result of many more TCP connections established.

TCP idle time: Per TCP flow idle times shown in Figure 5(c) are usually spent parsing the HTML content, executing Javascript, or performing other client processing before another request can be issued. Chosen empirically, one second threshold is used for excluding RTT-induced gaps. Similarly, server may be busy processing before replying with data. HTC/T-mobile is found to have the longest idle time explaining its longest page load time and smallest average throughput as shown in Figures 4(b)(c). The idle time ranking does not directly correlate with the page load time, partly due to effect of throughput and also as a result of large subsecond packet interarrival time gaps (*e.g.*, for HTC/AT&T).

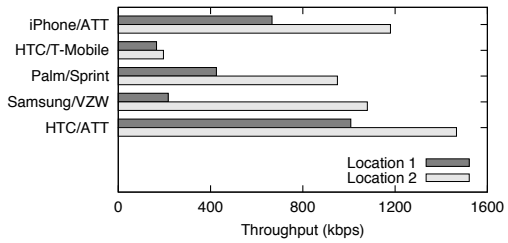


Fig. 6. Throughput difference at two locations.

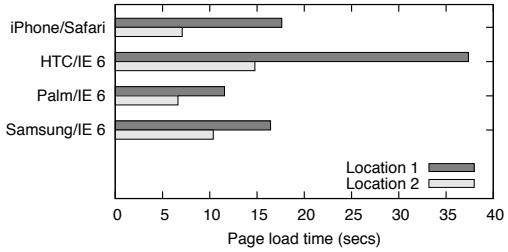


Fig. 7. Page load time difference at two locations.

TCP transfer: Per flow TCP transfer time distribution correlates reasonably well with the throughput behavior, as iPhone-WiFi-Proxy setup has shortest transfer time, while HTC/T-mobile is the worst.

It is important to note that fair comparison across platforms by visiting actual Web sites is impossible due to platform-specific content customization as indicated by large differences in the payload size (Figure 4(f)), and the actual IPs contacted for each URL visit. Moreover, differences in browser configuration may also impact the performance. For example, the maximum number of concurrent TCP connections allowed for each IP or domain may differ, as observed in Table 4, where iPhone is shown to have a much larger limit, accounting for its superior Web browsing performance.

6.2.2 Location dependence

For this study, all the experiments unless mentioned otherwise are carried out at a single location; however, it is known that location has an impact on data cellular network performance due resource and usage differences. Here we show the influence of location over the performance of transport layer and application layer in Figure 6 and 7 using TCP downlink throughput and Web page load time.

Figure 6 reflects the location on difference in TCP downlink throughput. A big performance gap exists between two places: In Loc2, the TCP downlink throughput is consistently better than that in Loc1. Samsung/VZW's downlink throughput is roughly 5x better, and Palm/Sprint and iPhone/ATT perform roughly 2x better. Although Loc2 performs much better than Loc1 in transport layer, in application layer Loc2 is only

slightly better indicating that application layer performance depends on not only transport layer capacity but also other factors.

Result summary: We have demonstrated the diverse application performance behavior for video streaming and Web browsing that do not directly correlate with the TCP/UDP throughput. Due to the complexity of Web browsing, it is difficult to fairly compare by visiting actual Web sites. Thus, we investigate the use of controlled experiments to dissect Web browsing performance.

7 Controlled Web Experiments

Given the difficulty caused by client-based content customization for narrowing down the bottleneck (*e.g.,* CPU for Javascript processing vs. network bottleneck for content download), visiting commercial Web sites alone is insufficient. We design a set of controlled experiments to ensure the same content downloaded for each URL visit and the same browser setting by creating a local Web server using Apache that replicates or mirrors the actual Web content of the sites studied previously. We collect altogether 1758 Web objects (including types such as images, applications, and text) associated with 128 distinct domain names and 258 server IPs. We noted previously that each browser has its own default setting for maximum number of allowed concurrent connections shown in Table 4. To ensure fair comparison we configure the maximum TCP concurrency on the client side to 12 which is the upper bound that observed in the trace.

Besides the server load behavior, our setup can fairly accurately represent the actual user experience, despite slightly different network paths traversed. The reason is that we expect the bottleneck to reside in the 3G/2G network or on the devices which are mostly unchanged in the controlled experiments. We discuss next how each key contributing factor for Web performance previously examined can be effectively analyzed and moreover fairly compared across platforms, in addition to fine-grained Javascript execution and Web content parsing analysis.

DNS lookup: For each phone setup, a small program is run to perform lookups for all 128 unique domain names. The distribution of per DNS lookup completion time is shown in Figure 8(a), which actually disagrees in the rank order shown in Figure 5(a), explained by the observation that a different set of names are resolved for each platform. Understanding DNS delays provides insight into the usefulness of DNS caching.

TCP handshake: Similar to DNS experiments, each platform establishes a TCP connection with each of the 258 server IPs from the trace in sequence to obtain the TCP handshake time distribution shown in Figure 8(b) where some curves crossing each other, indicating an inconsistent rank order. Compared to Figure 5(b), the rank order also differs and surprisingly iPhone's performance appears to be slightly worse for some TCP connections likely due to worse network performance.

Web object downloading: Popular Web sites often contain many embedded web objects in the main page, including images, Javascripts, CSS files, *etc*. Although Web object download and execution can happen concurrently, it is still important to measure web object download time separately which could be a bottleneck during web browsing. We measured the download time of all 1758 observed Web objects by sending the same HTTP GET requests found in the trace. Figure 8(c) shows the distribution, where iPhone-WiFi-Proxy setup again is much superior due to higher network bandwidth. T-Mobile has longest download time as it is using a 2G network. Confirming previous observations, HTC/AT&T again shows much worse performance compared to iPhone/AT&T setup. This confirms our previous conjecture that device-based traffic differentiation exists in some 3G networks.

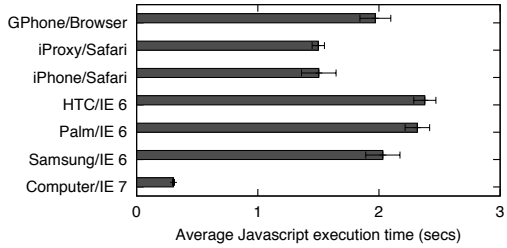
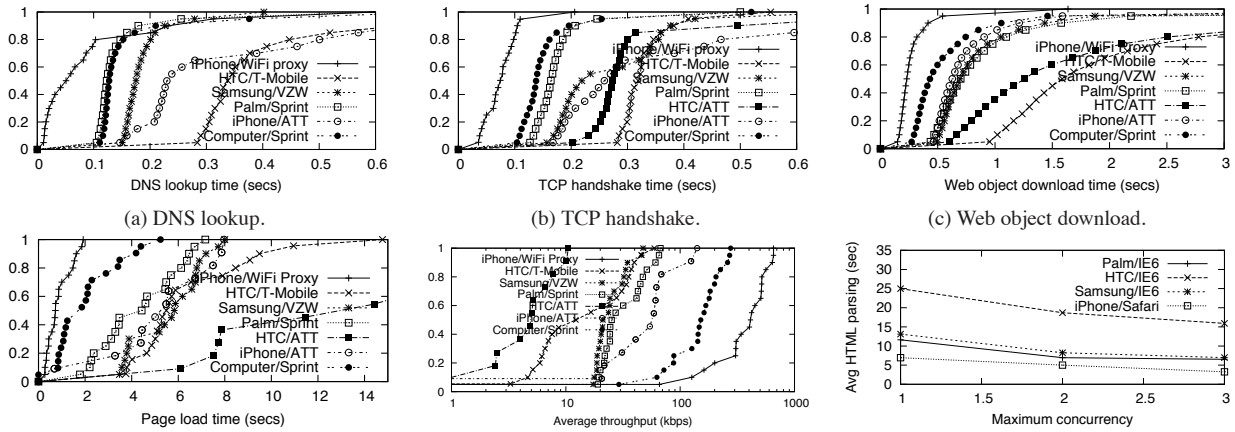


Fig. 9. Controlled experiments for JS execution.

Performance: We measure page load time and throughput to compare the performance of Web browsing across platforms as shown in Figures 8(d)(e) respectively. We did not plot the idle time distribution, as we found very few gaps lasting longer than 1 sec, likely due to shorter round-trip delay, lighter server load, and generally faster networks in the wired part of the testbed infrastructure compared to experiments visiting commercial Web sites. Consistent with previous observations, HTC/AT&T again is an outlier in experiencing worst performance, outperformed even by T-Mobile's 2G network. Computer/Sprint setup immediately trails behind the iPhone-WiFi-Proxy setup with the best performance, proving the importance of high processing capabilities. The average throughput for iPhone/AT&T is among the third best, demonstrating sufficient network resources within the AT&T 3G networks.

Concurrent TCP connections: Previously, we found iPhone has somewhat an unfair advantage, as we found it is configured with a much higher limit for maximum number of allowed concurrent TCP connections for each server IP or domain. We also explore how increasing maximum TCP concurrency, conveniently configured at our server side due to a lack of control for iPhone, has an impact on the average HTML parsing time. This metric is measured as the elapsed between the start of the first byte from the server until the end of the last HTTP request which is embedded at the end of the Web page using a Javascript. Overlapping with parsing, this time also includes download time which is improved with increasing concurrency as shown in Figure 8(f).

Javascript execution time: Due to limited resources,



(a) DNS lookup. (b) TCP handshake. (c) Web object download.

(d) Page load time. (e) Average throughput. (f) Concurrency's effect on HTML parsing.

Fig. 8. Controlled experiments for Web transaction breakdown.

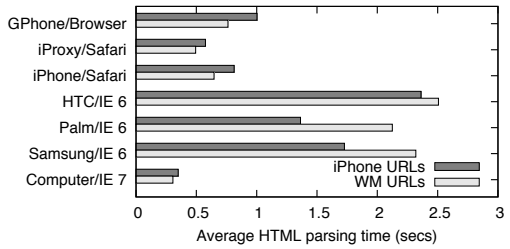


Fig. 10. Controlled experiments for HTML parsing.

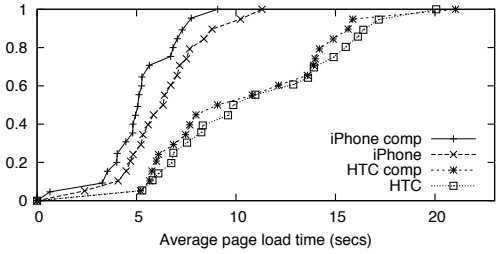


Fig. 11. Page load time comparison (iPhone vs. HTC, compressed vs. uncompressed).

it takes browsers on smartphones more time to parse and render Web objects compared to browsers executing on desktop computers. We design a set of controlled experiments to understand HTML parsing overhead and Javascript execution time.

For Javascript analysis, we extract all embedded Javascript objects from real network traces of visiting the 24 Web sites. By replaying to Web servers the same HTTP GET requests in the trace, we can download all Javascript objects on a desktop computer. Two sets Javascripts are obtained: one corresponding to iPhone's web browsing, the other for Windows Mobile phones. To ensure fair comparison, we exclude platform-specific Javascripts which first infer the browser type and then branch accordingly. Though it is hard to precisely identify such Javascripts, we apply heuristics by searching key words such as *ActiveXObject*, indicative of being IE-specific. 40 scripts are collected after filtering based on a list of key words.

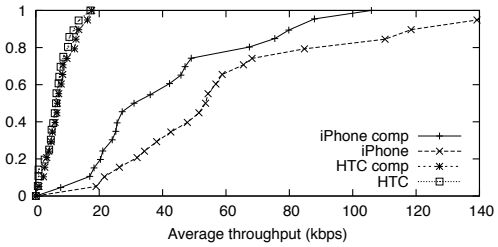


Fig. 12. Throughput comparison (iPhone vs. HTC, compressed vs. uncompressed).

To measure Javascript execution time, we first create an HTML page which includes all selected scripts. Two HTTP HEAD requests are used to mark the start and end of the execution of all scripts. This is to bypass a bug in the implementation of Date objects for IE browsers on some phones preventing us directly invoking getMilliseconds(). HEAD requests are used for correlating the start and end time of the execution in the packet traces. A small drawback of this method is that the time in the packet trace of the HTTP HEAD request packet always appears later than the timestamp of xmlhttp.send(null), which is used to initiate HTTP HEAD request due to scheduling. However, this time gap of less than 100 ms is negligible relative to the aggregate Javascript execution time and HTML parsing time on the order of seconds.

To ascertain to what extent browsers on smartphones are slower than the desktop counterparts, we repeat these experiments using Internet Explorer 7.0 on a desktop machine running Windows Vista. The result is shown in Figure 9 which demonstrates the clear advantage of computer based platform over smartphones, among which iPhone performs the best.

HTML parsing time: To assess the HTML parsing overhead, we modify the mirrored Web pages on our local Web server to contain a simple line of Javascript to initiate an HTTP request at two locations: one right af-

ter <*body*> and the other is right before </*html*>. From the packet traces, we can identify the time difference between these two HTTP requests. This is is a good estimate for the HTML parsing time, as most of the HTML parsing time is spent on parsing its <*body*> element and the delay of sending out HTTP request packet caused by scheduling is insignificant.

In this set of experiments, all phones and computer use WiFi connection so that network conditions are the same across all platforms. For all browsers, the maximum TCP connections per domain is set to 20, large enough to not affect the experiment. Figure 10 shows the HTML parsing performance for two sets of URLs: iPhone URLs containing content downloaded using iPhone and WM URLs including content obtained on Windows Mobile browsers. iPhone-based setups have performance quite close to that of computer, GPhone coming second, followed by HTC, Palm, and Samsung which have similar performance.

Compression: Aside from content customization, another factor that is difficult to control in real browsing experiments is whether the content is compressed. Such setting is configured at the server. Compression reduces the data size, but incurs additional processing at the device due to the need for decompression. We investigate the benefit of compression across platforms by configuring our servers with two modes: always sending compressed content vs. sending only uncompressed content. Figures 11 and 12 illustrate the page load time and throughput respectively across setups for both compressed and uncompressed content. We observe that for the same AT&T network, HTC consistently performs much worse than iPhone or any other platform, but not due to compression overhead. The performance improvement due to compression is negligible, *i.e.,* iPhone's page load time is improved by only 1 second on average.

Result summary: We have demonstrated the benefit of using controlled experiments to dissect and fairly compare the contributing factors for Web browsing performance across platforms. Our analysis has shown iPhone's clear advantage in Javascript execution and HTML parsing speed, and higher average throughput. Desktop platforms also exhibit much higher performance in execution and render of Web content. We can conclude that network is still likely the main bottleneck for pages we have characterized. For example, Javascript execution time for 40 scripts is at the same order of as the transfer time for only one Web object.

In particular, we note the benefit of using locally available network infrastructure by connecting smartphones to a desktop with wired connectivity through an Ad-Hoc WiFi network using the desktop as a proxy to access network services. We explored the setup for iPhone (iProxy) and have shown its superior performance over other setups due to higher bandwidth and smaller RTT values while accessing the same content as 3G network.

8 Conclusion

In this paper we characterized the performance of network applications on smartphones in a way that is relevant to end-users, cellular operators, and smartphone vendors. Our goal was to provide users with data and analysis that equips them to make an informed decision about which carrier is good for their specific needs. We conducted detailed analysis of application performance along several dimensions that are of interest to cellular network operator and hardware and software vendors. This analysis provides guidance on how they can improve their networks and devices. Most importantly, we presented a systematic black-box methodology for measuring performance of cellular data networks from the perspective of end-users and application developers. We believe our results are an important step towards understanding of cellular networks and smartphones.

9 References

[1] Broadband DSLReports.com. `http://www.dslreports.com/archive`.

[2] Apple Takes 2nd in Smartphone Market Share, But Q4 Looking Good for RIM. http://www.intomobile.com/2008/11/09/apple-takes-2nd-in-smartphone-market-share-but-q4-looking-good-for-rim.html, November 2008.

[3] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt. Performance Optimizations for Wireless Wide-Area Networks: Comparative Study and Experimental Evaluation. In *Proceedings of ACM MOBICOM*, 2004.

[4] J. Chesterfield, R. Chakravorty, J. Crowcroft, P. Rodriguez, and S. Banerjee. "Experiences with multimedia streaming over 2.5G and 3G networks". *Journal ACM/MONET*, 2004.

[5] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *IMC '07*, 2007.

[6] M. Ghaderi, A. Sridharan, H. Zang, D. Towsley, and R. Cruz. Modeling tcp in a multi-rate multi-user cdma system. In *IFIP-Networking 2007*, 2007.

[7] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3g network: Interplay between the wireless channel and applications. In *Proceedings of ACM MOBICOM*, September 2008.

[8] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering Performance Differences in Backbone ISPs with Netdiff. In *Proceeding of NSDI*, 2008.

[9] K. Mattar, A. Sridharan, H. Zang, I. Matta, and A. Bestavros. Tcp over cdma2000 networks : A cross-layer measurement study. In *PAM*, 2007.

[10] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, 1998.

[11] P. Rodriguez, S. Rangarajan, and S. Mukherjee. Session Level Techniques for Improving Web Browsing Performance over Wireless Links. In *Proceedings of World Wide Web*, 2004.

[12] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley. Inference and evaluation of split-connection approaches in cellular data networks. In *PAM*, 2006.

[13] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz. Primary users in cellular networks: A large-scale measurement study. In *DySpAN*, 2008.

[14] Z. Zhuang, T.-Y. Chang, R. Sivakumar, and A. Velayutham. A3: Application-Aware Acceleration for Wireless Data Networks. In *Proc. of ACM MOBICOM*, 2006.