Measurement Approaches to Evaluate Performance Optimizations for Wide-Area Wireless Networks

Rajiv Chakravorty[1], Julian Chesterfield[1], Pablo Rodriguez[2] and Suman Banerjee[3]

[1] University of Cambridge Computer Laboratory, Cambridge CB3 0FD, UK
{firstname.lastname}@cl.cam.ac.uk
[2] Microsoft Research, Cambridge CB3 0FD, UK
pablo@microsoft.com
[3] University of Wisconsin, Madison WI 53706, USA
suman@cs.wisc.edu

Abstract. We present measurement approaches to evaluate performance optimizations, employed at different layers of the protocol stack, to enhance application performance over wide-area wireless networks (WWANs). Applications running over WWAN cellular environments (e.g web browsing) are significantly affected by the vagaries of the cellular wireless links. Much of the prior research has focussed on variety of isolated performance optimizations and their measurements over wired and wireless environments. In this paper we introduce experiment-based measurement approaches to benchmark application performance using optimizations performed at individual layers of the protocol stack.

These measurement initiatives are aimed at: (1) performing an accurate benchmark of application performance over commercially deployed WWAN environments, (2) characterizing the impact of a wide selection of optimization techniques applied at different layers of the protocol stack, and (3) quantifying the interdependencies between the different optimization techniques and providing measurement initiatives for future experimentation to obtain consistent and repeatable application benchmarks in WWAN environments.

1 Introduction

All over the world wide-area wireless networks (WWANs) are being upgraded to support 2.5G and 3G mobile data services. Unfortunately, application performance over WWANs is severely impacted by problems of the cellular wireless medium – high and variable round trip times, fluctuating bandwidths, frequent link outages, burst losses, etc. [1]. As a consequence, the end-user experience in such environments is significantly different from the relatively stable indoor wireless environments, e.g. 802.11b based Wireless LANs (WLANs).

In this paper, we consider measurement approaches and performance benchmarks over WWANs from an end-user perspective. Our performance study explores questions like:

- *What measurement approaches can yield reproducible and repeatable (web-browsing) experiments over WWANs?*
- *What factors contribute to the poor application performance (web) over WWANs?*
- *What different optimizations can be applied at individual layers of the protocol stack and what is the benefit available from each?*

To answer these questions, we have conducted an empirical performance study involving real WWAN networks and applications. Our approach differs from other approaches and work conducted over WWANs. While previous research has investigated large-scale performance study of end-to-end TCP flows [5] and also its cross-layer interaction with the link-layer [3, 4], the approach taken in this paper differs from them in several ways. First, we quantify the causes of poor application performance and quantify real *user experience* over WWANs. We accurately measure the different components that contribute to the latencies during web downloads for a range of popular websites (ranked in `www.100hot.com`). Second, we introduce *virtual web hosting* as an important construct to perform repeatable web browsing experiments over WWANs. Third, we benchmark all standard web browsers, protocols, and techniques with respect to their performance. Finally, we implement and study a wide selection of optimization techniques at different layers and their cross layer interactions on application performance.

Our paper is laid out as follows. The next section describes our experimental WWAN infrastructure and elaborates on our methodology to conduct repeatable and reproducible experiments over WWANs. In section 3, we discuss some of our empirical findings for optimizations applied at different layers of the protocol stack. Section 4 discusses our results while the last section concludes the paper.

2 Testbed Infrastructure and Methodology

We focussed our experimental evaluation on web-browsing performance over a WWAN testbed. We used a commercial GPRS-based WWAN network as shown in figure 1. In this testbed, the mobile terminal (MT), e.g. a laptop, connects to the GPRS network through a GPRS-enabled interface – a PCMCIA GPRS card or a phone. In our experiments the MT (or mobile client) downloaded web content over the WWAN link from different content locations: (1) directly from the real web servers, e.g. CNN, Yahoo, and (2) virtually hosted web-servers (explained later in this section) that were located in our laboratory.

To study the different optimization techniques at different layers of the protocol stack and their overall impact on application (web) performance, our experiments also required us to implement optimization-specific proxies. Based on the use of proxies, our experiments were classified into three modes:

- **No Proxy Mode:** In this case the client directly connected to the server and the experiments did not require any intervening proxy. These optimizations are the easiest to deploy.

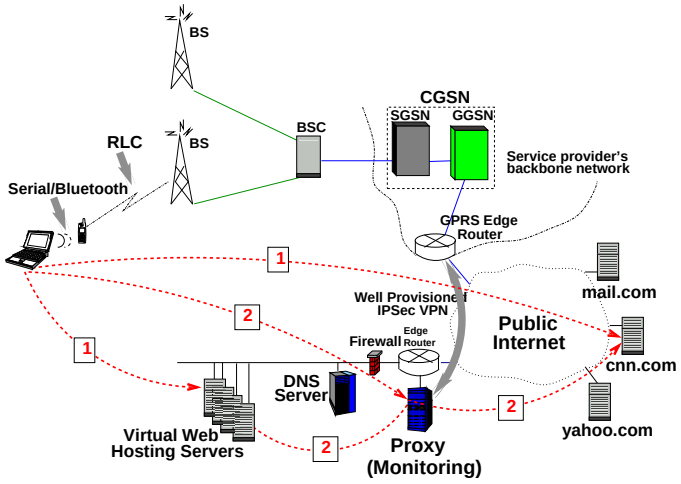

Fig. 1. WWAN Architecture and Testbed. In these experiments, we placed the proxy in our laboratory and then use a well provisioned IPSec VPN to 'back haul' GPRS traffic to it directly from the cellular provider's network. In the proxy-based experiments, the mobile client connects to the web servers through this proxy (Label 2 in Figure 1).

- **Transparent Proxy Mode:** This mode is used for those experiments where the client need not be aware of the existence of a proxy and the cellular provider's network *transparently* guides the client's connections through a proxy as necessary.
- **Explicit Proxy Mode:** This mode was used in experiments which require the mobile client to be aware of the proxy in the network (in this case called the 'server-side' proxy). This requires either, (a) explicit browser configuration or (b) software update at the mobile client to make it appropriately interact with the server-side proxy. The software update is like a client-side proxy and hence we refer to this approach as a 'dual-proxy' solution.

Furthermore, in our experiments we have used *virtual web hosting* to emulate real web downloads. Virtual web hosting is an important construct to perform repeatable web browsing experiments over WWAN links involving typically fast-changing websites.

Why Virtual Web Hosting? Contents of popular websites change very frequently (e.g. in CNN content changes within minutes). If real web-download experiments were to be conducted over low-bandwidth WWAN links involving such web-sites, then different download attempts may notice significant differences in the downloaded content structure and volume. In other words, the total time to perform each set of experiments for every individual website was much higher than the time it takes for the web-page content to change. Hence, it would not have been feasible for us to make meaningful comparisons performed directly

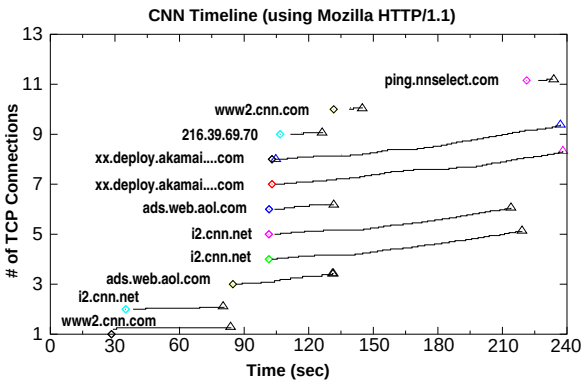

Fig. 2. Timeline for an example web download over WWAN networks, using Mozilla/HTTP/1.1. The web content is spread over 6 servers and multiple connections are opened by the browser to these servers. As the HTTP/1.1 default behavior dictates, only two simultaneous TCP connections are opened to a specific server. Each small rise in the lines indicates a separate GET request made using that specific connection.

using real websites. To avoid this problem we implemented a *virtual web hosting* system in our laboratory, where we statically replicated the contents of the popular websites into a set of web servers (hosted in our laboratory) that were made publicly accessible for the mobile client. Thus, a mobile client can access the virtually hosted webpages using WWAN networks just as they would from the real servers *in a repeatable and reproducible fashion*.

Replicating Distributed Web Content for WWANs. Web downloads of popular websites such as `www.cnn.com` access a number of distinct domains spread across multiple CDN servers, e.g., Akamai, to download content (see figure 2). This access pattern significantly affects the download performance over WWAN links for two reasons: (1) number of DNS look-ups, and, (2) number of TCP connections opened by the client to these different servers, etc. To emulate this aspect of web downloads in the virtual web hosting setup, it was necessary to faithfully replicate the distributed web content and its overall structure. For each server in the original website, we assigned a separate web server in our laboratory to "virtually" host the corresponding content. The domain names of these virtual-hosting servers were constructed from their original domain names by pre-pending the corresponding CDN server domain names. These modified domain names were made available to the DNS. Additionally, we updated the URLs pointing to the embedded content to reflect the new domain names. Thus, in a virtual web hosting experiment when a mobile client attempts to download a webpage, it would have to appropriately resolve different domain names for the different content servers similar to the case of a real web download.

Website	Download latency (sec) WWAN-Real	WWAN-Virtual	No. of Dom.	Emb. Objects (Size in KB) Count	Sum	Avg.	Max.	T'put(Kb/s)
mail	43.3 (5.5)	34.5 (3.4)	4	11	36.7	3.3	11.0	**8.5**
yahoo	38.8 (4.1)	35.0 (3.1)	6	16	60.3	3.8	36.0	**13.8**
amazon	102.3 (9.8)	76.4 (7.7)	3	42	91.9	2.2	46.8	**9.6**
cnn	204.0 (17.6)	196.3 (12.4)	6	67	186.8	2.8	22.3	**7.6**

Table 1. Web Download latencies (using Mozilla/HTTP/1.1) and other characteristics for 4 popular websites and their content distribution. During experiments, mobile host was stationary at a location with reasonably good link conditions (e.g. typical C/I > 15dB). HTTP 1.1 achieves *abysmally* low average throughputs over WWANs.

Our experiments performed using virtual web hosting replicate the key components of the web browsing performance that any WWAN user would experience with actual web servers. However, there exists few differences between overall performance observed using the real web servers and virtual web hosting scenario. We have observed that the the mean download latencies are lower (by about 5-10%) for the virtual-hosting system. This is primarily due to the (1) absence of dynamically generated content, (2) difference in server workload and processing times in the virtual-hosting case. We emphasize that none of the above performance differences change the qualitative nature of the results when comparing the different optimization techniques.

3 Experiences with Performance Optimizations

Our experimental evaluation is focused on web-browsing performance over a WWAN network. We have experimented with different standard web-browsers available (e.g. Mozilla, Internet Explorer, Netscape). Though there are minor variations in their implementations, we observed that their performance is roughly similar. Our results show that the default configuration parameters of most browsers (typically chosen to work well in wired networks or wireless LANs) perform poorly in WWAN environments. This is surprising in the context of prior work [4], which showed that TCP, the underlying transport protocol used by HTTP, makes efficient use of the WWAN wireless link. Our results also show that individual TCP connections are relatively efficient over these wireless links. However, the HTTP protocol needs to be suitably adapted to improve its performance over WWAN environments.

In order to precisely benchmark web performance, we have used the Mozilla browser version 1.4. In its default setting Mozilla opens upto 8 simultaneous TCP connections per web-server using HTTP 1.0 and upto 2 TCP connections using HTTP/1.1, Mozilla also supports proposed experimental features in HTTP/1.1, e.g. pipelining.

File Size (KB)	FTP-throughput (Kbps)
1	13.2 (1.5)
5	18.1 (0.9)
10	18.8 (2.1)
50	29.7 (3.3)
100	30.5 (3.2)

Table 2. Data throughputs achieved for ftp-downloads over WWAN wireless links using a single TCP connection. TCP achieves good throughputs for larger files.

3.1 Performance Benchmarks

We conducted experiments for a number of different websites and we briefly summarize four of them in Table 1. These four websites were chosen based on the diversity of their characteristics, content types, content volumes, and number of servers used. The download latencies of the different websites have significant variability due to the diversity in content and the multiplicity of servers. The table also indicates the overall data throughput achieved in downloading these websites. We can observe that the overall throughput is significantly low. It varies between only 7.5 Kbps to 17 Kbps for different websites, even though the ideal downlink data-rate is 39.6 Kbps. We can contrast the performance of this web download to ftp-like data transfers presented in Table 2. In this table we present the throughput achieved when we downloaded a single file (of different sizes) over the same WWAN wireless link.

The throughput achieved in such file transfer experiments were significantly higher than the web downloads. For example the web download throughput for `amazon.com` with a total content size of 91.9 KB was 9.6 Kbps, while the download of a single 50 or 100 KB file was around 30 Kbps! The high file transfer data throughput confirms prior observations made by Ludwig et. al. [4] that TCP performs quite well over GSM-based wireless links. This implies that there are significant inefficiencies in the web download mechanisms and carefully applied optimizations can significantly improve the performance.

3.2 Performance Optimizations

We have examined the performance of a wide-selection of optimization techniques that have been proposed at the different layers of the protocol stack — application, session, transport, and link. As discussed in Section 2 some of these optimization techniques relied on a transparent or explicit proxy that was located in our laboratory. In this section we will discuss the benefits observed by each of these techniques, except for the explicit dual-proxy techniques in most cases. The dual-proxy techniques works with very different assumptions of deployment and hence it is not possible to make a fair comparison of these techniques with the no-proxy or single-proxy techniques. Therefore, we will only comment on the benefits of the schemes individually and their combined effects in the summary

of results (Section 4). We now discuss performance optimizations.

Application layer Optimizations. For application layer optimizations, we quantified the benefits of schemes like HTTP pipelining, extended caching, delta encoding, and dynamic content compression.

Dynamic Data Compression. We implemented dynamic content compression using an application-level proxy operating in the transparent as well as the explicit dual-proxy mode. From our experiments, we have observed that the content in the different websites are very compressible. However, the benefits of compression on application performance may not be as substantial (except for the case of Yahoo). This apparent anomalous behavior is due to the typical object size distribution of some webpages. Here we observe that most of the objects in the webpages can be small, e.g. nearly 60% of the objects in a CNN snapshot were less than 1 KB (typically 1 TCP segment, assuming 1460 byte payloads of IP packets). Any amount of compression would clearly not change the number of segments below one. Therefore the overheads of issuing individual GET requests for these objects sequentially over the two TCP connections dominates the transfer time of these objects and hence the improvement in data transfer latency due to compression will be minimal in these cases. In contrast, for web sites where the distribution of object sizes is skewed towards larger values (e.g. Yahoo) the impact on download latencies is higher.

HTTP Pipelining. We evaluated performance of the HTTP 1.1 protocol. The default persistent HTTP/1.1 protocol gets each of these small objects sequentially over its two TCP connections, and waits numerous times between the completion of each GET request and the beginning of the next. In contrast, HTTP pipelining allows many GET requests to be issued simultaneously by the mobile client and hence the objects are fetched without any intervening gaps. From our experiments, we see that HTTP pipelining provides between 35% to 56% benefits for the different websites. HTTP pipelining is an an experimental technique in the HTTP/1.1 standard and we found that, unfortunately, most browsers do not enable this feature by default.

CHK-based Caching/Delta Compression. We also investigated performance of extended CHK-based caching and delta coding for different web-sites. Our experiments show that such techniques on average improves real web-browsing experience by about 3-6% for fast-changing web-sites.

Session level Optimizations. We performed a detailed study of performance enhancement schemes like (1) the impact of multiple simultaneous transport connections as typical in standard web browsers, (2) impact of DNS look-ups on web downloads [2], and, (3) parse-and-push technique.

Varying TCP Connections. We investigated an alternative session layer technique to optimally choose the number of simultaneous TCP connections opened by the client to the server. We found that for a base capacity of the GPRS handset (39.6 Kbps in our case) increasing the number of TCP connections (from 2

to 6) leads to significant improvement in the user experience (i.e. for CNN the download latency reduces from 196.3 seconds to 123.0 seconds).

DNS Boosting. DNS-boosting achieves the same effect as URL re-writing (as in Content Distribution Networks) by intelligently manipulating DNS queries from the client. Specific details of this scheme is available in [2]. Note that this can significantly benefit performance in two ways: (1) by avoiding extra DNS Lookups and (2) by reducing the number of TCP connections opened by a web browser. We implemented this technique as a proxy and performed download experiments for the different websites. By eliminating the DNS lookups for the transparent proxy, we achieve another 5-9% improvement in the download latency. The net improvements due to the session and application techniques are between 53-65%.

Parse-n-Push. Parse-and-push is a session-level, explicit, dual-proxy scheme that emulates deterministic content pushing towards the mobile client, when the wireless downlink would have been otherwise left idle. While supporting parse-and-push mechanism requires explicit client-side software update, the scheme helps to improve overall utilization of the link. Our experiments have shown that Parse-and-push provides an additional 5%-12% improvement in the web download latency for the popular websites.

Transport layer Optimizations. We evaluated the performance of standard TCP, a recently proposed link-adapted variant suited for WWAN environments (TCP-WWAN), and a customized UDP based transport (UDP-GPRS) solution.

Using two different proxies, we quantified the additional benefits of using link-adapted TCP and custom UDP based solution. In these experiments, we apply the application-level optimizations (full compression) and session-level optimizations. We have observed that using TCP-WWAN (transparently deployed) achieves between 5-13% additional benefits for the different websites. UDP-GPRS custom protocol (dual-proxy approach) leverages its specific knowledge of the wireless link characteristics to improve the download performance further (between 7-14% for the different websites).

Link layer Optimizations. Using trace-based simulations, we have studied the interaction between link-layer retransmissions (ARQ) and forward error correction (FEC) schemes in WWAN environments. We have investigated mechanisms that allow the RLC to dynamically choose the encoding schemes in conjunction with the ability to enable or disable ARQ, and the impact of such mechanisms on applications. Performing actual experimentation for this study was difficult since we had no control on the encoding schemes used by the Base Station to transmit data the mobile client. At the mobile client we only had the flexibility to enable or disable ARQ, and the ability to disable FECs. Therefore, we performed trace-based simulations to study the data performance for various applications over a wide range of channel conditions and encoding choices.

Our study confirms that for each different channel condition there is an optimal value of FEC that leads to the least download latency. For example a

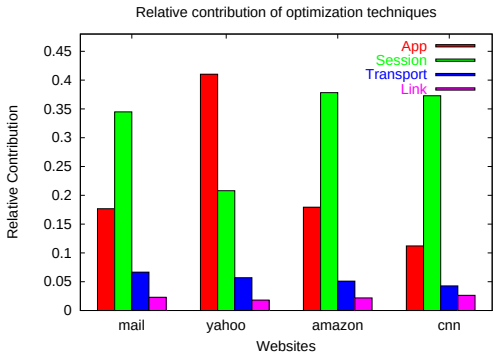

Fig. 3. Relative contribution of optimizations for 4 popular web-sites.

moderately poor channel, with an error rate of 0.9% on the GPRS channel, 5-6% FEC is the optimal choice to minimize download times. The amount of required FEC for such optimal performance increases with increase in channel error rates. This suggests that the RLC should continuously monitor the channel conditions and *dynamically* choose the amount of FEC to be applied on reliable data transfers across the wireless links.

4 Summary of Results

From the experiments conducted, we classified the performance optimizations into two classes — those that require re-configuration or software update in the mobile client, i.e. uses an explicit proxy and those that have no such requirements. Here we assumed a reasonably good wireless link (error less than 0.2%) where dynamic (adaptive) FECs provide a latency improvement of about 5%. This value is derived from our link-layer trace-based simulations to improve performance.

In Figure 3, we only plot the relative contribution of the schemes that require no client-side reconfiguration when all optimizations *are applied simultaneously.* For example, in Amazon application, session, transport, and link layer optimization techniques contribute 17.9%, 37.8%, 5.1%, and 2.2% respectively. The improvement provided by all the techniques applied simultaneously were the sum of these values, 63.0%, which brought the download latency for Amazon from 76.4 seconds to 29.3 seconds. ***In general, we can observe that application and session layer techniques have a dominating effect in improving the web performance.*** They lead to 48-61% web performance improvements for our example websites. Thus our work demonstrates that the application and session-level mechanisms currently deployed for web browsing applications make poor use of the relatively efficient lower layers. Employing appropriate optimizations at these layers (as described in this paper) can help bridging this performance gap observed between the upper and lower layers. Our results show the

benefits to be somewhat higher when client-side reconfiguration/software update is applied.

Note that transport, and link layers optimizations typically provide an additional 5-10% performance improvement (considering reasonably good link conditions), which is still significant for web downloads over WWAN links.

5 Conclusions and Ongoing Work

Preliminary results from our comparative performance study of different optimization techniques reveals the following: (1) There is a significant mismatch in the performance of default HTTP protocols and its underlying transport mechanism, TCP. Unlike wireline networks, standard web browsers are unable to exploit even the meagre resources of the WWAN links. (2) Significant performance benefits can be realized by suitable optimizations implemented at the application and session layers. Commercial web servers and browsers should implement the HTTP-pipelining scheme, which provides noticeable benefits to end-user performance. (3) Inspite of significant compressibility of web content, dynamic data compression techniques do not provide commensurate performance benefits. (4) Custom protocols, explicitly designed for WWAN environments, present significant performance benefits at the transport layer. However, in many cases the deployment of such schemes can be expensive for the service providers.

In our ongoing work, we are conducting more thorough experiments including range of other popular web-sites to obtain even more accurate web browsing benchmarks. We are also investigating other novel approaches for benchmarking application performance across realistic web server workloads and in presence of dynamically changing web content. We plan to extend this study for other WWANs e.g. UMTS, CDMA 2000.

References

1. R. Chakravorty and I. Pratt.: "Performance Issues with General Packet Radio Service", *Journal of Communications and Networks (JCN)*, Vol. 4, No. 2, December 2002.
2. P. Rodriguez and S. Mukherjee and S. Rangarajan.: "Session-level techniques to Improve Web Browsing Performance over Wide-Area Wireless Links", *Proc. of the World Wide Web (WWW) Conference, 2004* (to appear).
3. M. Meyer.: "TCP Performance over GPRS", *Proc. of IEEE WCNC 1999*.
4. R. Ludwig, et al.: "Multi-Layer Tracing of TCP over a Reliable Wireless Link", *Proc. of ACM SIGMETRICS 1999*.
5. P. Benko, et al.: "A Large-scale, Passive Analysis of End-to-End TCP Performance over GPRS", *Proc. of the IEEE INFOCOM 2004 (to appear)*.