Performance Optimizations for Wireless Wide-area Networks: Comparative Study and Experimental Evaluation

Rajiv Chakravorty, Suman Banerjee, Julian Chesterfield, Pablo Rodriguez, Ian Pratt *

ABSTRACT

We present a comparative performance study of a wide selection of optimization techniques to enhance application performance in the context of wide-area wireless networks (WWANs). Unlike in traditional wired and wireless IP-based networks, applications running over WWAN cellular environments are significantly affected by the vagaries of the cellular wireless medium. Prior research has proposed and analyzed optimizations at individual layers of the protocol stack. In contrast, we introduce the first detailed experiment-based evaluation and comparison of all such optimization techniques in a commercial WWAN testbed. This paper, therefore, summarizes our experience in implementing and deploying an infrastructure to improve WWAN performance.

The goals of this paper are: (1) to perform an accurate benchmark of application performance over such commercially deployed WWAN environments, (2) to implement and characterize the impact of various optimization techniques across different layers of the protocol stack, and (3) to quantify their interdependencies in realistic scenarios. Additionally, we discuss measurement pitfalls that we experienced and provide guidelines that may be useful for future experimentation in WWAN environments.

1. INTRODUCTION

Wireless cellular networks are being upgraded world-wide to support 2.5G and 3G mobile data services. For example, GPRS and UMTS networks in Europe, and CDMA 1xRTT and CDMA 2000 networks in the USA and Asia are currently being deployed and tested to provide wireless data services that enable ubiquitous mobile access to IP-based applications. In this paper we examine the performance of these IP-based applications running over such Wireless Wide-Area Networks (WWANs).

WWAN links are severely plagued with problems like high round trip times (RTT), fluctuating and relatively low bandwidths, frequent link outages and burst losses [14, 31, 33]. As a consequence the end-user experience in the WWAN environment is significantly different from the relatively stable indoor wireless environments, e.g. 802.11b based Wireless LANs (WLANs). Over the past several years, various optimizations have been proposed to different layers of WWAN protocol stacks to improve the experience of data users. Our work presents a detailed comparative study of different proposed performance optimization techniques through experiments over commercial cellular networks. We believe that ours is the first study of its kind that presents a detailed comparison of a wide-selection of such optimization techniques across all layers of the protocol stack, studies the cross-layer interactions of these techniques and their impact *on application performance* in a realistic experimental environment consisting of *commercial cellular wireless networks.*

We have conducted detailed performance studies of different applications over commercial WWAN networks. In this paper, we primarily focus on the performance of web browsing applications in these environments. Our experiments show that all standard web browsers operating in their default settings *significantly* under-utilize the limited resources of the WWAN wireless link. This is surprising in the context of prior work [31], which showed that TCP, the underlying transport protocol used by HTTP, makes efficient use of the WWAN wireless link. In fact, these results of [31] are also re-confirmed by our experiments. In this paper we explain this performance discrepancy based on inefficiencies in session and application layers. Our experiments show that suitable optimizations implemented in these layers can significantly improve application performance over WWAN environments.

Overview

To precisely quantify the causes of poor performance over WWANs, we first benchmarked standard web browsers, protocols, and techniques with respect to their performance. Through our experiments we have measured the different components that contribute to the latencies during web downloads for a range of popular websites (ranked in www.100hot.com). Subsequently, we examined a large number of optimization choices that are available at different layers of the protocol

*R. Chakravorty, J. Chesterfield, I. Pratt are with the Computer Laboratory, University of Cambridge. Emails: {rajiv.chakravorty, julian.chesterfield, ian.pratt}@cl.cam.ac.uk. S. Banerjee is with the Department of Computer Sciences, University of Wisconsin-Madison. Email: suman@cs.wisc.edu. P. Rodriguez is with Microsofft Research, Cambridge. Email: pablo@microsoft.com.

.

stack. Specifically we study the following aspects of these optimizations for WWANs in this paper:

- **Application layer:** We quantify the benefits of using schemes like HTTP pipelining [27, 29], extended hash-based caching, delta encoding [26] and dynamic content compression over WWANs.

- **Session layer:** We study the impact of multiple simultaneous transport connections as typical in standard web browsers, examine the impact of techniques like DNS-boosting/URL-rewriting [32] and of server-side 'parse-and-push' [12].

- **Transport layer:** We evaluate the performance of standard TCP, a recently proposed and implemented link-adapted TCP variant suited for WWAN environments [13] and a customized UDP based solution [12].

- **Link layer:** We study the interaction between link-layer retransmissions (ARQ) and forward error correction (FEC) schemes using trace-based simulations for different applications in WWAN environments.

To conduct this study we implemented all the necessary techniques (including three different proxies) for our WWAN infrastructure. The work presented in this paper summarizes our experiences and lessons learnt through deployment and operation of our WWAN testbed over a 12 month period.

The following were some of our interesting observations: (1) Although TCP itself is relatively efficient even in WWAN environments, the default HTTP protocol significantly under-utilizes the WWAN wireless links. (2) Appropriate application-layer mechanisms are necessary to correct the significant mismatch between transport and application performance. (3) Proxy-based optimizations are sometimes crucial to realize many of the performance benefits.

Key Contributions

We highlight the main contribution of this work as follows:

- We present the first detailed evaluation of *application performance over commercial WWAN environments.*

- We implement and study a wide selection of optimization techniques *at different layers* and their cross layer interactions on application performance.

- We present an experiment methodology based on *virtual web hosting* that is essential for performing reproducible and repeatable experiments over WWAN environments.

Additionally, to encourage further measurement and evaluation studies within our research community, we are in the process of releasing our tools and software which implements the optimization techniques in Win2000/XP and Linux, detailed traces, and benchmarks used.

Roadmap

The rest of this paper is organized as follows. In the next section we describe our testbed and experimental methodology. In Section 3 we present a benchmark of the existing web browsing performance using standard browsers, protocols, and techniques over WWAN environments. In Section 4 we present a detailed evaluation of various optimization techniques and their relative benefits to application performance. In Section 5 we discuss some of the deployment issues for these techniques. In Section 6 we present some of the related work and finally conclude in Section 7.

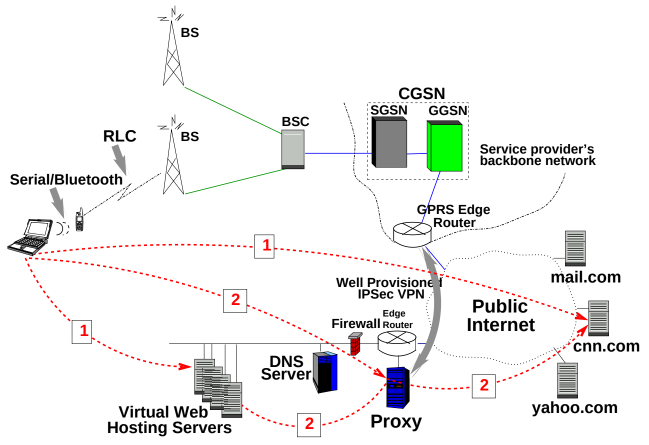

Figure 1: WWAN Experimental Testbed. In our experiments, we placed the proxy in our laboratory and then use a well provisioned IPSec VPN to 'back haul' the traffic from the cellular provider's network. The mobile client connects to the web servers through this proxy (shown as Label 2).

2. TESTBED AND METHODOLOGY

The experiments reported in this paper have primarily been performed on GPRS-based WWAN networks, which are widely deployed in different parts of the world. Figure 1 shows a commercial GPRS-based WWAN network that was used in our experiments. GPRS [9] is a bearer service for GSM, and two new nodes have been added to the traditional GSM network to support GPRS: Serving GPRS Support Node (SGSN) and Gateway GPRS Service Node (GGSN). The SGSN acts as a packet switch that performs signaling similar to a mobile switching center (MSC) in GSM networks, along with cell selection, routing, and handovers between different Base Switching Centers (BSCs). It also controls the Mobile Terminal (MT)'s access to the GPRS network and routes packets to the appropriate BSC. The GGSN is the gateway between the mobile packet routing of GPRS and the fixed IP routing of the Internet.

We have also experimentally evaluated application performance on the next generation (3G) cellular networks, which are currently being deployed in parts of North America and Asia. Wireless links using these 3G technlogies have higher data rates than existing 2.5G technologies. Our preliminary study over 3G networks (based on CDMA 2000 3G-1X) are presented in Section 5. These results indicate that our observations and evaluations presented in this paper apply equally to these higher data rate 3G environments.

2.1 Experimental and Infrastructure Setup

The experimental testbed we have used is shown in Figure 1. In this setup, a mobile terminal (MT), e.g. a laptop, connects to the WWAN network through a mobile device – a PCMCIA GPRS card or a phone. In order to use the WWAN network, the MT first *attaches* itself to the GGSN (PDSN in CDMA 2000) through a signaling procedure and establishes a Point-to-Point Protocol (PPP) connection with the GGSN. The MT is dynamically assigned an IP address and the WWAN network is responsible for switching data back and forth to this IP address as the MT moves through the network.

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. This is shown by Label 1 in Figure 1. In either case, the data from the mobile client to the servers traverse through the cellular service-provider's network as well as the public Internet, before it is finally 'back-hauled' to 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 required us to implement optimization-specific proxies. Based on the use of proxies, our experiments can be 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.
- **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. Transparent proxy solutions are also easy to deploy, since they require no changes or configuration to be made in the mobile clients themselves.
- **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 updates at the mobile client to enable it to 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.

In the proxy-based optimizations the proxy needs to be deployed within the cellular provider's network. Since it was not practical for us to install our own equipment inside a commercial cellular network, we instead placed the proxy in our laboratory and used a well provisioned IPSec VPN to 'back haul' GPRS traffic to it directly from the cellular provider's network. This ensured that the path between the proxy and the cellular provider's network was never the bottleneck. In the proxy-based experiments, the mobile client connects to the web servers through this proxy (Label 2 in Figure 1).

2.2 Experiment Methodology

We use 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 fast-changing websites.

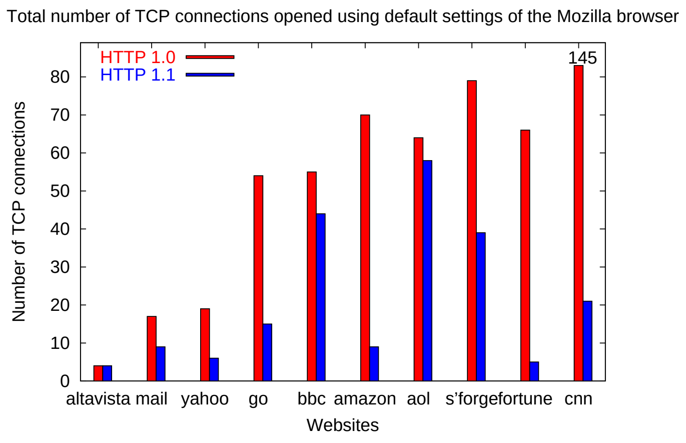

Figure 2: Total number of TCP connections opened by the client to download the main webpages of the popular websites with HTTP/1.0 and HTTP/1.1 protocols. Some of the web servers send 'pre-emptive FINs' and hence benefits of HTTP/1.1 are not always realized.

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. The total number of downloads required for each website considered in this paper was more than 500 (including at least 20 downloads for each configuration). This translates to a duration of over 1500 minutes to perform the experiments, leaving aside setup times, transient network congestion, unavailable wireless link conditions, etc. Hence it would not have been feasible for us to make meaningful comparisons performed directly using real websites. To avoid this problem we implemented a *virtual web hosting* system in our laboratory, where we replicated the contents of the popular websites into a set of web servers (in our laboratory) with public domain names. A mobile client can access these webpages using WWAN networks just as they would from the actual servers.

A snapshot on the main webpage of a popular news website like CNN shows that it has more than 100 embedded objects. While some of these objects may be hosted on the main content servers, others are hosted by CDN servers, e.g. Akamai [22]. (as shown in Figure 3).

When a user attempts to download `http://www.cnn.com`, the browser first performs a DNS lookup to resolve `cnn.com` and downloads the main webpage. Subsequently it performs further DNS lookups to resolve the CDN servers that hold some of the embedded objects in that page, and performs appropriate downloads for these objects. Downloads from popular websites sometimes involves a large number of DNS lookups, which significantly affect the download performance over WWAN links. Hence, in the virtual web hosting setup, it was necessary to faithfully replicate the distributed web content and its overall structure.

	Download latency (sec)				No. of	Embedded Objects (Size in KB)				**T'put(Kbps)**
Website	WWAN-Real	WWAN-Virtual	Wired	802.11	Domains	Count	Sum	Avg.	Max.	(Virtual)
altavista	14.2 (3.1)	13.3 (2.2)	0.7	1.1	2	4	13.6	3.4	10.1	**8.2**
mail	43.3 (5.5)	34.5 (3.4)	1.0	1.3	4	11	36.7	3.3	11.0	**8.5**
yahoo	38.8 (4.1)	35.0 (3.1)	1.1	1.5	6	16	60.3	3.8	36.0	**13.8**
go	90.3 (11.5)	69.2 (4.5)	2.0	2.8	3	29	92.8	3.2	51.8	**10.7**
bbc	93.7 (9.1)	72.6 (6.1)	2.1	3.0	2	35	72.5	2.1	38.4	**8.0**
amazon	102.3 (9.8)	76.4 (7.7)	2.1	2.9	3	42	91.9	2.2	46.8	**9.6**
aol	105.5 (11.5)	80.0 (7.3)	2.7	3.4	8	66	168.5	2.6	21.7	**16.9**
sourceforge	107.2 (11.7)	92.1 (6.1)	2.6	2.9	5	53	132.3	2.5	40.3	**11.5**
fortunecity	121.3 (9.0)	114.4 (7.1)	3.0	3.6	1	46	168.7	3.7	59.1	**11.8**
cnn	204.0 (17.6)	196.3 (12.4)	3.1	4.2	6	67	186.8	2.8	22.3	**7.6**

Table 1: Web download latencies (using Mozilla/HTTP/1.1) over GPRS WWAN and other characteristics for different websites and their content distribution. Standard deviations for the download times of Wired and 802.11 were low and not shown. In these tests, mobile host was stationary with reasonably good link conditions (average measured C/I > 15dB).

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 web-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.

The experiments performed using virtual web hosting replicate the key components of the real web browsing performance in the same manner as any WWAN user would experience with actual web servers. However, there are a few differences between overall performance observed using the real web servers and the virtual web hosting scenario:

- **Server-side Load:** The load at the real web servers may be different than at the web servers in virtual web hosting scenario. Typically real web servers experience significant variation in the user load. However, we were interested in quantifying the differences due to protocol techniques and not the server load, and hence this was particularly beneficial for us.

- **Pre-emptive Server FINs:** Web downloads using the HTTP/1.1 protocol [29] make use of persistent connections, i.e. multiple objects from a specific server are downloaded using the same TCP connection. Padmanabhan and Mogul in [28] first showed that such a behavior can significantly improve the web download performance over the original behavior of the HTTP/1.0 protocol, where each object is downloaded by a separate TCP connection. However we have observed that some popular web servers explicitly close TCP connections by sending a pre-emptive FIN after having transmitted a few objects over that connection. Consequently the client has to open a new connection to resume downloading the remaining objects in the webpage. In some cases, web servers perform such 'early-close' to avoid holding state for too many outstanding TCP connections, thereby improving server performance [6] and also guarding against possible DDoS attacks. We illustrate this behavior in Figure 2 which shows the total number of TCP connections required to download the main webpage of 10 popular websites. When the HTTP/1.0 protocol is used each client needs to open a large number of TCP connections to download all embedded objects of a webpage. When using HTTP/1.1, if the web servers faithfully implement persistent connections, then the number of opened connections should vastly reduce, as is the case for Go, Fortunecity, etc. However, for the websites that employ pre-emptive server FINs (even when the client uses HTTP `keep-alive`), e.g. BBC, AOL, the total number of TCP connections opened by the HTTP/1.0 and HTTP/1.1 versions differ by less than 20%. Clearly, some of the advantages of using HTTP/1.1 protocol are lost for web servers that employ pre-emptive FINs. In experiments involving web servers in a virtual web hosting scenario, we chose to adhere strictly to the HTTP/1.1 protocol. Thus, web downloads using web servers in our virtual web hosting scenario, will not explicitly close (FIN) TCP connections.

- **Web Content:** Apart from static content, many web servers dynamically generate additional content which is downloaded by the clients. It is difficult to exactly replicate such behavior in the virtual web hosting scenario. In many cases the total number of objects and the amount of content downloaded from a virtually-hosted website is lower than the actual website.

We emphasize that none of the above performance differences change the qualitative nature of the results when comparing the different optimization techniques. In fact, there is significant value in using the more controlled virtual web-hosting setup to study application performance over WWAN environments in Table 1.

In the table we illustrate some of these above differences. WWAN-Real corresponds to downloads from the real servers of some popular websites, while WWAN-Virtual corresponds to downloads from our virtual-hosting system (the values in

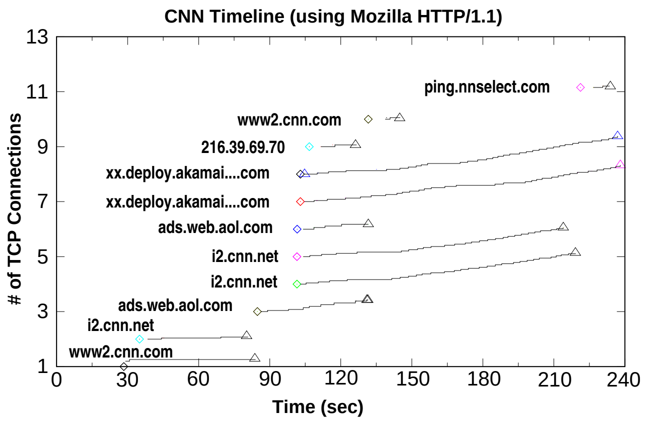

Figure 3: Timeline for an example web download over WWAN networks, using Mozilla/HTTP/1.1. The web content is spread over 6 servers and multiple connections are opened by the browser to these servers. As the HTTP/1.1 default behavior dictates, only two simultaneous TCP connections are opened to a specific server. Each small rise in the lines indicates a separate GET request made using that specific connection.

parenthesis are the standard deviations). We can observe that the downloads from real websites experience a much higher variation, most of which are due to frequent changes in content and varying server loads. Additionally we can observe that the mean download latencies is lower (by 5-10%) for the virtual-hosting system. This is because of the absence of the dynamically generated content in the virtual-hosting case. Finally, the columns labeled Wired and 802.11 list the corresponding web download times over 100 Mbps Ethernet and 11 Mbps 802.11-based wireless LAN environments. Clearly, the download performance in such environments is orders of magnitude better. In fact, the key difference between these wired and wireless LAN environments to that of WWAN environments is the high and variable Round Trip Times (RTTs) seen on the latter. For example, the first packet in a burst of back-to-back WWAN packets experience WWAN link latencies in the range of 600 ms to 3 seconds under different channel conditions. Based on these observations we can conclude that the net effect of the differences in virtual web hosting scenario from the actual one is marginal, when compared to the overwhelming impact of the WWAN environment itself.

We now focus on our main application, wireless web browsing, and quantify the causes of its poor performance over WWAN wireless links.

3. BENCHMARKING PERFORMANCE

Our experimental evaluation is focused on the 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. In the experimental results reported in this paper, we use 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. Due to its open source nature, it was easier to alter the default browser to suit experimental needs. Our experiments reported in this paper were performed using a laptop with a 1.4 GHz processor running Linux (kernel 2.4.20) which connects to the WWAN network using a '3+1' GPRS phone, which has an ideal downlink channel data rate of 39.6 Kbps. Unless otherwise stated, all experiments were repeated 20 or more times and typically report the mean and standard deviations, in parenthesis, of these values. (In some tables we omit the standard deviation values for ease of exposition and clarity.)

In Figure 3 we plot an example timeline to download `http://www.cnn.com` when using the HTTP/1.1 protocol. The plot shows the progress of a download with time, the cumulative count for the number of TCP connections opened, the duration of these connections, and the extent of data download activity through these connections over the WWAN network.

We ran similar experiments for a large number of different websites which we summarize in Table 1. 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 files 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 in similar network conditions 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. [31] and Benko et.al. [8] that TCP performs quite well over GSM-based wireless links.

Note that such under-performance of web browsing applications is not unique to GPRS-based 2.5G cellular networks alone. As presented in Section 5, even 3G wireless links like CDMA 2000 3G1X exhibit similar traits. We performed experiments on a CDMA 2000 network using a phone with ideal downlink data rate of 144 Kbps. TCP itself achieves a data throughput between 95-125 Kbps [14]. In contrast the web download time for Yahoo's main webpage (total data volume of about 60.3 KB in this case) was around 12 seconds, i.e., a data throughput of mere 41 Kbps.

While we have performed a detailed experimental evaluation using all websites listed in Table 1, in the rest of this paper we present experimental results for four of them for the sake of clarity and of exposition. These four websites were chosen based on the diversity of their characteristics, content types, content volumes, and number of servers used. They were a news web-site (`cnn.com` with 186.8 KB over 6 servers), an online e-commerce store (`amazon.com` 91.9 KB over 3 servers), a popular mail web-site (`mail.com` 36.7 KB

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 throughput for larger files.

and 4 servers) and search portal (`yahoo.com` 60.3 KB and 4 servers).

To further understand the performance bottlenecks in web downloads, we partition the download latency over WWAN links for these four websites into constituent factors. We did this by obtaining and analyzing appropriate `tcpdump` traces at the mobile client as well at the web servers.

- **Payload Transfer Time:** This component accounts for the time when some useful data transfer is taking place over the WWAN link.
- **DNS lookups:** This accounts for the latency that is incurred over the WWAN link for different DNS lookups during downloads. (Note that more look-ups lead to more 'idle-time' instances.)
- **TCP 3W/SS:** This component accounts for the under-utilization of WWAN links during TCP's initial 3-way handshake and slow start phases.
- **Other under utilization:** This accounts for all the other link-layer and protocol inefficiencies, which can lead to under-utilization. The overall effect is that link is never able to achieve the expected data rate due to several reasons – physical (e.g. handover) and environmental (e.g. fading) effects, protocol under-performance (e.g. HTTP GET and TCP congestion control) etc.

In the plot we observe that for the smaller websites (e.g. mail.com) DNS lookup latency, TCP's initial 3-way handshake and slow start contributes to a significant fraction of the total latency incurred, e.g., about 29% for mail.com. These delays are, however, amortized over the downloads for the larger websites. The websites are sorted in the increasing order of their content size, and we can see a clear progression of this amortization as the content increases in size.

More importantly we can observe that a WWAN link is under-utilized for a large fraction of the download varying between 35% to 55% for different websites.

A contributing factor of this under-utilization is the distribution of object sizes and the manner in which the HTTP protocol performs the downloads. A large fraction of the objects in popular web sites have very small sizes (e.g. CNN has more than 70% of the objects less than 2 KB in size). Therefore, using HTTP/1.1 in the default mode (with two TCP connection) leads to a 'stop-and-go' behavior — the client makes a GET request for an object, receives it completely, and then makes the next GET request. Such a behavior minimally impacts application performance in low-

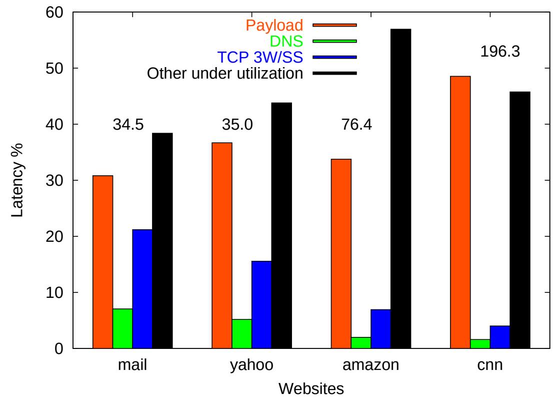

Figure 4: Distribution of latencies expended in various components of the download times for the main webpage of four of the Top-10 websites (as percentages of the entire download time). The numbers above the bars represent the actual time required to download the entire webpage, in seconds.

latency networks like 802.11-based wireless LANs, but leads to significant performance degradation in high-latency wireless links of the WWAN environments.

Other factors include physical channel fading effects, presence of large queues in typical WWAN networks and their impact round-trip time, and retransmission timeout inflation in TCP, etc.

These results indicate that there are significant opportunities for optimizing web browsing performance for WWAN networks.

4. PERFORMANCE OPTIMIZATIONS

We have examined and characterized 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 quantify the relative 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 present the benefits of the dual-proxy schemes individually and comment on their combined effects in the summary of results (Section 4.5).

4.1 Application Level Techniques

We consider three application-level optimization techniques. First, we examine content compression and its impact on web download performance. Our results show that web content is highly compressible, but this does not translate to commensurate benefits in web download performance. Next, we examine the various choices available through the HTTP protocol, e.g. HTTP/1.1 with persistent connections and HTTP request pipelining [27, 29]. 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. Finally, we explore some advanced techniques including delta encoding [26] and extended caching.

4.1.1 Dynamic Content Compression

The total size of content of many webpages is quite large relative to the downlink data rate of WWAN networks. Hence, content compression is a natural candidate to further reduce download latencies. We first examine the "compressibility" of the different websites. We classify content into two parts — fixed fidelity content (which includes all HTML, CSS, JS, files) and variable fidelity content (which includes all images). For all fixed fidelity data we applied *lossless* data compression, using `gzip` and for all variable fidelity data we applied *lossy* data compression, by reducing the depth level of images. Note that other forms of data compression, e.g. html reformatting, image resizing, etc. could also be applied here. However, we do not employ such techniques in our proxy, since they require significant knowledge of the content semantics. We implement dynamic content compression using an application-level proxy operating in the transparent as well as the explicit dual-proxy mode.

Website	Fixed-fidelity		Variable-fidelity		Total
	Orig.	Lossless	Orig.	Lossy	Compression
mail	11.8	3.0	17.4	8.8	59.6%
yahoo	36.8	7.1	24.9	11.7	69.5%
amazon	45.4	9.5	46.0	20.7	67.6%
cnn	160.0	65.3	35.4	27.6	52.6%

Table 3: Amount of compression achievable on the content of the four popular websites. For fixed fidelity content (HTML/CSS/JS objects) we apply lossless compression, and for variable fidelity content (jpeg/gif/bmp) we apply lossy compression). Data in KB.

In Table 3 we present the "compressibility" of the content in the four popular websites. All the content in the various website allow significant opportunities for data compression (between 52% to 70%).

Website	HTTP/1.1-def.	HTTP/1.1-def. + Compression			
	(2 conn.)	Lossy	Lossless	**Full**	Improv.
mail	34.5	32.6	31.9	28.4	**17.7%**
yahoo	35.1	32.2	23.6	20.7	**40.9%**
amazon	76.4	71.5	69.2	62.7	**17.9%**
cnn	196.3	187.5	183.1	174.3	**11.2%**

Table 4: Mean webpage download times improvements for dynamic content compression. Full compression implies both lossy and lossless compressions. Improvement with respect to HTTP/1.1-default that uses at most two simultaneous TCP connections (first column). Download time in seconds.

In Table 4 we present the performance benefits of content compression of web download latencies. When lossless compression is used, the client needed to perform an uncompress operation. (However the extra CPU overhead to uncompress, say a 100 KB data file, is of the order of a few milliseconds for most handheld devices and is insignificant.) We can observe that although the content in the different websites are very compressible, the benefits of compression on application performance is not as substantial (except for Yahoo). We can explain this apparent anomalous behavior based on the object size distribution of the webpages. We can observe that most of the objects in the webpages are small, e.g. nearly 60% of the objects in CNN are 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, the distribution of object sizes in the Yahoo website is skewed towards larger values (the average object size is 3.8KB in Yahoo, as opposed to 2.2 KB and 2.8 KB of Amazon and CNN respectively, listed in Table 1). Data compression has a significantly better impact on download latencies of such websites.

4.1.2 Optimizing HTTP using Pipelining

Many current web browsers continue to use non-persistent connections (HTTP/1.0), which opens-up new TCP connection for every object downloaded. In contrast, HTTP/1.1 in its default mode opens two TCP connections to each server which are used to download all the objects from that server. Our results in Figure 4 show that HTTP/1.1-default suffers from significant under-utilization of the WWAN link. Therefore we now study the impact of the HTTP/1.1-Pipelining feature which is an experimental option in the standard [29]. In Table 5 we summarize the different variants of the HTTP protocol that we study in this section and Section 4.2.1.

A browser that implements HTTP request pipelining is allowed to issue new GET requests without waiting for the entire response of the previous ones. Hence a browser can use this mechanism to issue simultaneous GET requests and ensure that the TCP connections are fully utilized for data transfer. This is in contrast to the HTTP/1.1-default (non-pipelined) which issues GET requests sequentially over each open TCP connection. In fact, it is precisely this 'stop-and-go' GET behavior of HTTP/1.1-default which leads to significant of under-utilization of the WWAN link.

Mechanism	Description
HTTP/1.0	Use of separate TCP conn. for each object downloaded
HTTP/1.1-def.	Use of two "persistent" TCP conn. to download all objects.
HTTP/1.1-Pipelining	Use of 2 "persistent" TCP conn. with simultaneous GETs (Sec. 4.1.2)
HTTP/1.1-Opt.	Use of 6 "persistent" TCP conn. (Sec. 4.2.1)

Table 5: Annotation of different HTTP mechanisms being discussed in this paper.

In Table 6 we can see that HTTP Pipelining provides between 35% to 56% benefit for the different websites. The benefit is particularly high for large websites like CNN with many objects in their webpage. We explain this using the

Website	Default (2 conn.)	**Pipel.**	Improv.	**Opt.** (6 conn.)	Improv.
mail	34.5	15.2	**55.9%**	21.5	**37.7%**
yahoo	35.1	22.7	**35.3%**	24.2	**31.1%**
amazon	76.4	39.2	**48.7%**	43.5	**43.1%**
cnn	196.3	89.1	**54.6%**	123.0	**37.3%**

Table 6: Mean webpage download times improvements for optimizations to browser configurations. Opt. is HTTP/1.1-Opt. (Sec. *4.2.1*). Pipel. is HTTP/1.1-Pipelining (Sec. *4.1.2*). Improvement with respect to the HTTP/1.1-default (2 connections).

distribution of object sizes shown earlier in Table 1. Most of the objects in these popular webpages have a large number of small objects (e.g. CNN has more than 60% of the objects less than 1 KB in size). The default 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, pipelining allows many GET requests to be issued simultaneously by the mobile client and hence the objects are fetched without any intervening gaps.

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. Additionally our experiments show that the web servers of many popular websites currently do not respect this pipelining feature. Therefore, the real performance benefits of this mechanism on wireless web browsing can be quite limited. In all our subsequent experimentation, we will ignore HTTP/1.1-pipelining technique for this reason and re-visit it in our summary of results (Section 4.5). In Section *4.2.1* we will study the impact of a session layer technique in which we vary the number of simultaneous TCP connections and how it can approximate the behavior of HTTP pipelining.

4.1.3 Extended Caching and Delta Encoding

In this explicit, dual-proxy based optimization, we evaluated extended content-hash based cachingand delta encoding schemes [26].

In this scheme, a 'client-proxy' software was installed in the mobile device which interacted with the server-side proxy located on the WWAN infrastructure as shown in Figure 1. In the extended caching scheme, the client as well as the server proxy indexes web-objects by their `SHA-1` fingerprint, which we call Content Hash Key (CHK). Each time the client attempts to download a webpage, it gets a CHK list of all the objects in the webpage from the server-side proxy. Using this information, the client makes request for only single instance of objects, even if they point to multiple URLs (same response but aliased to multiple URLs). Experiments have shown that in many dynamically generated websites (e.g. `bbc.co.uk`) such a phenomenon is commonplace (also known as response aliasing [21]). Hence, CHK-based caching is able to eliminate redundant data transfer over WWAN, save disk space and also improve overall web download times.

When the data being downloaded is a different version of the same object previously cached (i.e. same URL but different CHK), a delta encoding scheme is used. Delta encoding is a standard technique in which the server (in our case the server-side proxy) sends only the differences between the new and old versions of a document to the client. This technique is very useful for websites like `cnn.com` and `bbc.co.uk`, where the content changes incrementally, but frequently. Our experience shows that the use of CHK-based caching and delta-encoding on average improves real web-browsing performance by about 3-6% in such fast-changing web-sites.

4.2 Session-layer Techniques

The goal of the session-layer optimizations is to mitigate the link 'idle time' effects incurred during DNS lookups and some other factors during web downloads. We performed a detailed study of the performance enhancement schemes for: (1) impact of multiple simultaneous transport connections as typical in standard web browsers, (2) impact of DNS look-ups on web downloads [32], and, (3) server 'parse-and-push' technique. We study URL-rewriting/DNS-boosting with the transparent proxy, and Parse-and-Push scheme as an explicit dual-proxy approach.

4.2.1 Varying Number of TCP Connections

In Section *4.1.2* we showed that the HTTP pipelining feature has significant impact on web browsing performance in WWAN environments. Unfortunately, the pipelining feature is not faithfully supported by many commercial web servers. Therefore we look at an alternative session layer technique where we treat each HTTP download as a single session and optimally choose the number of simultaneous TCP connections opened by the client to the server.

Figure 5 shows how the normalized web-download performance varied with number of simultaneous TCP connections in use. The normalization was done with respect to HTTP/1.1-default (two connections). In all the cases, we found the best performance when 6 connections were used (we call this HTTP/1.1-Opt). The download latency for HTTP/1.1-Opt reduced by 35-42% for the different websites. The number of parallel GET operations increase with the number of connections, thus, decreasing the total 'idle time' on the wireless channel. However, we observe no additional benefits of increasing the number of connections beyond six, at which point the download performance is limited by other inefficiencies (e.g. DNS lookups, TCP 3-way handshake and slow start effects, etc.). In fact the performance degrades due to interaction between the TCP connections.

The optimal number of TCP connections that minimizes download latency is a property of the wireless interface and the network. Our results indicate that 6 TCP connections provide maximum throughput when using a GRPS network with an ideal downlink data rate of 39.6 Kbps. The optimal number of TCP connections can vary for a CDMA 2000 network with an ideal downlink data rate of 144 Kbps, and we recommend an empirical configuration of the browser settings based on such properties of the WWAN network.

Increasing the number of simultaneous TCP connections is an aggressive behavior. However, for bandwidth-depleted environments like WWAN environments, such a behavior leads to significant improvement in the user experience (i.e. for CNN the download latency reduces from 196.3 seconds to 123.0 seconds).

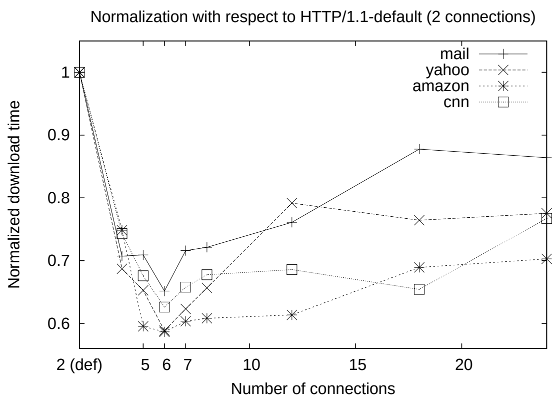

Figure 5: Download performance as the number of simultaneous TCP connections was varied. Download times normalized with respect to the HTTP/1.1 default of two TCP connections.

In Table 6 we can observe that the use of HTTP/1.1-Opt (6 connections) leads to significant performance benefits for web downloads. While the performance improvement of HTTP/1.1-Opt is similar to that of HTTP/1.1-Pipelining, it is somewhat lower in some cases. This is because HTTP/1.1-Opt is only able to approximate the behavior of HTTP/1.1-Pipelining by carefully choosing the number of simultaneous TCP connections.

The goal of both these optimizations, in the WWAN context, are similar — to reduce the number and overall duration of idle times between successive GET requests over each TCP connection. Since most web servers currently do not support the HTTP pipelining option, HTTP/1.1-Opt is an alternative approximation that may be used by WWAN clients to realize similar benefits.

4.2.2 URL-rewriting/DNS-boosting

These are two almost equivalent techniques that *transparently* reduce the DNS lookup overheads at the mobile client [32]. In the URL re-writing technique, the proxy in the cellular provider's network intercepts the GET request issued by the client for a webpage, say `index.html`, and makes appropriate server requests on the client's behalf. On receiving the `index.html` page, the proxy parses the contents, and identifies the *names* of all other servers that holds different embedded objects and replaces them with *its own IP address* similar to schemes employed by Content Distribution Networks [22]. It then responds with this modified `index.html` to the client. The client now directly contacts the proxy using the latter's IP address for these embedded objects through subsequent GET requests. The proxy again (pre-)fetches these objects from the other servers and in turn serves them to the client, much like a web caching system. Thus the client needs to perform at most one DNS lookup.

DNS-boosting achieves the same effect as URL re-writing, but by intelligently manipulating DNS queries from the client. By responding to the DNS queries with a *fixed IP address*, the DNS-booting technique implicitly forces the client to point to one single server so that client can open optimal number of TCP connections thereby improving overall performance. Thus, we find that both schemes can benefit performance in two ways: (1) by avoiding extra DNS Lookups, and, (2) by using optimal choice of TCP connections opened by a web browser to the server. More detailed description of both the schemes is available in [32].

	Includes App. Opts: Full Compression			
Website	1.1-def.	1.1-Opt only	1.1-Opt + DNS-b/URL-r	Improv.
mail	28.4	17.7	16.5	**58.1%**
yahoo	20.7	14.3	13.4	**64.7%**
amazon	62.7	35.7	33.8	**53.9%**
cnn	174.3	109.2	101.1	**58.0%**

Table 7: Additional benefits of session-level techniques. Download time in seconds. Improvement with respect to HTTP/1.1-def. (first column).

To quantify the benefits of these schemes, we implemented the DNS-boosting/URL-rewriting proxy and performed download experiments for different websites. We present the results in Table 7. Since we wanted to quantify the additional benefits of the different session level techniques, we performed the application-level optimizations (not including HTTP/1.1-pipelining since its prevalence is still limited in commercial web servers) in all these experiments. As noted before, use of HTTP/1.1-Opt itself leads to significant performance benefits, i.e. between 37-44%, while elimination of the DNS lookup overhead adds another 5-9% improvement in the download latency. The combined improvements due to these session layer techniques are between 53-65%.

4.2.3 Server-side Parse-and-Push

Parse-and-push is a session-level, explicit, dual-proxy scheme where the server-side proxy located at the other end of the wireless link in the WWAN network attempts to speculatively 'push' objects towards the client that it knows the client will have to download. For example, when the client makes a request for the `index.html` page, the server-side proxy will begin pushing the various objects embedded in the `index.html` file even before the client makes explicit GET requests for it. Parse-and-push 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 can provide an additional 5%-12% improvement in the web-page download latency for popular websites.

4.3 Transport-layer Techniques

We now examine two optimization techniques in the transport layer that have been proposed in recent literature [12, 13]. The first one is a transparent-proxy solution that attempts to optimize TCP performance using a 'transparent' proxy located in the cellular provider's network [13]. We refer to this technique as TCP-WWAN. The other is an 'explicit dual-proxy' solution which defines a custom protocol based on UDP [12] (we call it UDP-GPRS.) While TCP is designed to operate over a wide-range of network and link conditions, the optimized protocols studied in this section specifically leverage the knowledge of the underlying WWAN wireless links and hence achieve improved perfor-

Website	App. + Session Opts: 1.1-Opt + DNS-b/URL-r				
	None	T1	Improv.	T2	Improv.
mail	16.5	15.7	**11.2%**	14.2	**13.6%**
yahoo	13.4	11.6	**12.7%**	11.4	**14.2%**
amazon	33.8	30.8	**8.6%**	29.9	**11.3%**
cnn	101.1	96.24	**4.8%**	92.7	**7.7%**

Table 8: Benefits of transport-level optimizations techniques. Download time in seconds. None implies no transport optimizations, T1 is TCP-WWAN and T2 is UDP-GPRS. Improvements with respect to None (first column).

mance. Due to space constraints we only summarize the key features of both these implementations.

4.3.1 TCP WWAN

TCP WWAN defines a transparent proxy-based solution in which the proxy is located in the cellular provider's network [13]. It specifically addresses some of the main performance problems of TCP for web downloads over WWANs. For example, instead of using TCP slow start, it uses a pre-determined value of the bandwidth-delay product and performs aggressive recovery during packet losses and link stalls. Note that such aggressive behavior can be very disruptive if implemented in the Internet. However, TCP-WWAN is implemented only within the cellular provider's network which already implement appropriate bandwidth sharing mechanisms between users at lower layer of the protocol stack.

Standard TCP leads to large queue build-up due to generous buffer provisioning in most WWAN networks. This impacts TCP performance over WWAN links to be sometimes degraded due to *spurious timeouts*. When the wireless link 'stalls' occur, undelivered TCP segments accumulate in these large queues. However, once the link returns to normalcy, these segments are then correctly re-transmitted by the reliable WWAN link layer on link repair (causing 'delay spikes'), but the TCP sender timeouts because its retransmission timer has expired meanwhile. TCP-WWAN can estimate the available bandwidth on the wireless link and regulate the flow of TCP segments towards the mobile client such that such queue build-up is avoided inside WWAN networks. This prevents the TCP sender from spuriously timing out, when such wireless link stalls occur. We implemented a TCP-WWAN proxy for an experimental evaluation of this technique based on the description in [13].

4.3.2 Custom Transport Protocol

UDP-GPRS [12] is an explicit dual-proxy based scheme to improve the transport performance of web downloads. This scheme defines a reliable protocol using UDP and implements ordered, reliable, message transfer. The protocol is optimized specifically for GPRS networks by leveraging its knowledge of the GPRS wireless link. For example, this protocol is aware that the GPRS link layers offer reliable in-order data delivery. Hence it uses a selective repeat with Negative Acknowledgements for loss recovery. Using such specific properties and characteristics of GPRS links, this protocol responds efficiently even in the event of common patterns of packet losses. Periodic messages are generated every few seconds which allow hosts to detect serious link stalls. If such a link stall is detected, the client disconnects and re-attaches to the GPRS network. Experience has shown that this action often repairs the link failures. An unaware transport protocol (e.g. default TCP) will experience severe back-offs and failures under similar circumstances. UDP-GPRS, by design, also avoids TCP's connection setup and slow start delays. While TCP has to operate over links with widely varying qualities, being a custom solution UDP-GPRS can make many more assumptions about the underlying network. For instance, since GPRS networks implement a mechanism to share bandwidth between users, there is no need for the UDP-GPRS protocol to implement its own congestion avoidance mechanisms. Instead, it employs a simple credit-based flow control scheme. The credit value is so chosen to ensure that the wireless link remains fully utilized even though the buffer occupancy in the cellular network remains low. This avoids excess queueing that some long-lived TCP flows cause.

In Table 8, we present the additional performance benefits of applying the above transport-level optimizations. We can observe that TCP-WWAN achieves between 5-13% additional benefits for the different websites. UDP-GPRS leverages its specific knowledge of the wireless link characteristics to improve the download performance further (between 7-14% for the different websites).

4.4 Link-layer Techniques

We finally present an evaluation of link-layer mechanisms and their impact on user performance. WWAN wireless links use two different schemes to provide reliability across the wireless links over a wide range of channel noise conditions. The first of this is a data encoding scheme with various levels of Forward Error Correction (FEC). For example, GPRS networks use four different FEC schemes (CS-1 to CS-4) [9, 10]. The choice of the appropriate encoding scheme is made *statically* by the Radio Link Control (RLC) layer. Most current GPRS WWANs make use of the CS-2 scheme, which allows good data protection in moderate to high noisy radio conditions [20]. The second one is an Automatic Repeat Request (ARQ) scheme that works aggressively to recover any data transfer losses through re-transmissions. Since re-transmissions incur delays, all short term link outages are hidden from the higher layers and manifest as increased delays. The higher layers will detect losses only for (1) deep fading that leads to bursty losses, or (2) cell-reselection due to the cell update procedure that leads to 'black-outs'.

In this section we study mechanisms that will 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 all of 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. In order to study the tradeoffs between ARQ-based and variable FEC-based link layer reliability approaches, in some cases we relied on trace-based simulations (the traces were generated from actual experiments on our testbed).

Our traces were generated as follows: we sent a stream of UDP packets from the proxy (shown in Figure 1) to the mobile client. The WWAN wireless link is the bottleneck in

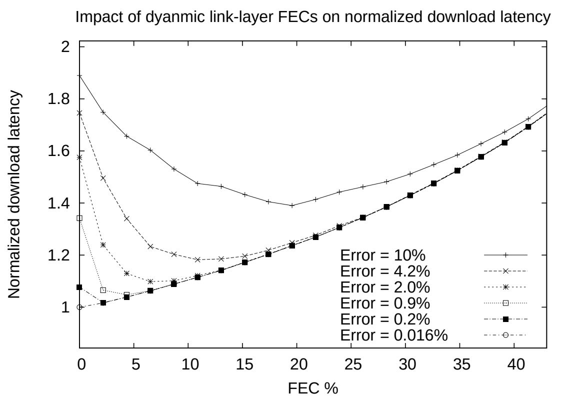

Figure 6: Impact of FEC-based link-layer data recovery for GPRS wireless links on bulk data downloads under different channel conditions (WWAN trace-driven simulations).

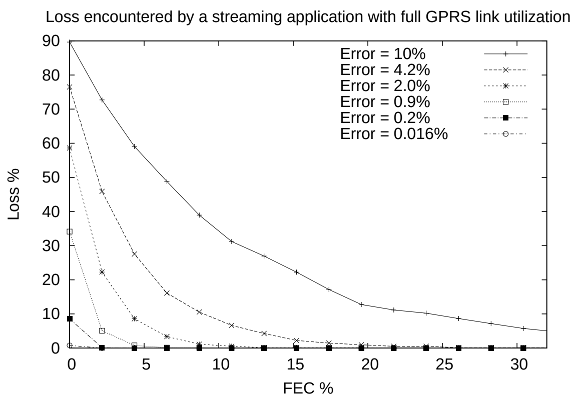

Figure 7: Impact of link-layer FECs on the data loss rate experienced by streaming applications (WWAN trace-driven simulations).

this system, and therefore, these UDP packets will queue at the base station. The RLC layer will appropriately fragment these UDP/IP packets into blocks and pace them out to the mobile client at the rate permissible by the wireless link. To generate our traces, we disable ARQ. Additionally we set the RLC layer at the mobile client to deliver all packets up to the IP layer (including packets in error) [1]. The RLC blocks are sub-divided into slots, and we can infer which slots are corrupted over the wireless link. We then apply various levels of FEC-based encodings and ARQ on these traces and observe their performance on the applications.

We consider two kinds of applications in this study: (a) reliable data transfer applications (like ftp and web traffic), and, (b) non-reliable data transfer applications (like streaming media). For reliable data transfer applications, we assume that the link layer can dynamically choose an amount of FEC (including none) to apply on the RLC data blocks. For ease of exposition, we assume that the FEC is applied at the granularity of slots. If there are 100 slots to a block and the amount of FEC applied is 5%, then 5 out of these 100 slots are used to redundantly encode the remaining 95. Slots that are not recovered after FEC is applied (due to higher channel error conditions) are recovered using the ARQ scheme. For the non-reliable data transfer applications we completely disable ARQ. Hence data that could not be recovered after FEC is applied, is lost.

4.4.1 Reliable Data Transfer Applications

In Figure 6, we plot the normalized data download latency for reliable data transfer for a large data file for different channel conditions and amount of FEC (the data is normalized with respect to the experiment with the best channel condition and no FEC). It is easy to see that for each different channel condition there is an optimal value of FEC that leads to the least download latency. For example a 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. By doing so, the data latency reduces by about 21.9%. In a better channel (say with 0.2% error rate) the corresponding benefit is about 5%. The amount of required FEC for such optimal performance increases with increase in channel error rates. This data suggests that, instead of using a fixed value of FEC (e.g. CS-2 in GPRS), networks should implement an RLC that can continuously monitor the channel conditions and dynamically choose the amount of FEC to be applied on reliable data transfers across the wireless links.

4.4.2 Non-Reliable Data Transfer Applications

An important consideration for non-reliable data transfer applications like media streaming is the jitter experienced by the data stream. Our experiments indicate that use of ARQ schemes for data recovery incurs a high jitter on traffic — it can vary between 600 ms to 3 seconds depending on the number of re-transmissions attempted. For such non-reliable data applications we consider the scenario where ARQ is completely disabled. Only variable amounts of FEC is used to recover from link layer losses. Use of FEC incurs constant delay overhead even when the channel is loss free. While this would decrease jitter between successive data packets, under high noise channel conditions it would also incur packet losses. In Figure 7 we show how the choice of FEC impacts losses experienced by a streaming application. We can see that even a low amount of FEC is sufficient to obtain a low loss performance for streaming applications, while maintain low jitter. However, a disadvantage of using FEC to handle losses is a reduction in data throughput. For example, when the channel error rate is 0.9%, 5-6% FEC is useful to eliminate most of the channel errors. However, this also leads to a reduction of useful data bandwidth by 5-6%. The gains of such FEC-based approaches are in corresponding reduction in jitter.

4.5 Summary of Results

In the previous sections, we presented a number of different optimization techniques to improve web download performance. We now summarize the main results. In this section, we assume a reasonably good wireless link, (error less than 0.2%) where dynamic FECs provide a latency improve-

[1]Blocks with corrupted headers could not not be correctly interpreted and hence was not delivered to the upper layers.

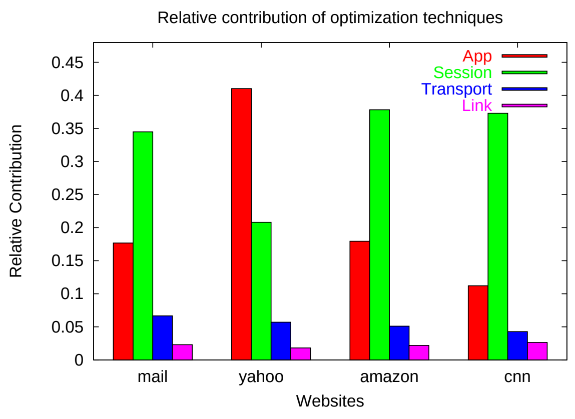

Figure 8: Relative contribution of optimizations at different levels of the protocol stack.

ment of upto 5%. This value is derived from our trace-based simulations. The benefits of dynamic FECs will increase with poorer wireless channel conditions and decrease with further improved channel conditions.

We have explored two classes of optimizations — those that require re-configuration or software update in the mobile client, i.e. uses an explicit proxy, (called *Client-reconf.*) and those that have no such requirements (called *No-reconf.*). In Figure 8, we plot the relative contribution of the No-reconf. schemes when all of them *are applied simultaneously*. The optimizations include, Full Compression, HTTP/1.1-Opt, DNS-boosting/URL-rewriting, TCP-WWAN, and dynamic FECs. For example, in Amazon application, session, transport, and link layer 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 from 76.4 seconds to 29.3 seconds. **In general, we can observe that application and session layer techniques dominate improvements in web performance. They lead to 48-61% 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 bridge this performance gap observed between the upper and lower layers.

Website	No-reconf.-I		No-reconf.-II		Client-reconf.	
	Lat.	Improv.	Lat.	Improv.	Lat.	Improv.
mail	15.7	**54.4%**	13.2	**61.6%**	12.4	**64.0%**
yahoo	11.6	**66.9%**	11.4	**67.2%**	9.9	**71.7%**
amazon	30.8	**59.6%**	27.4	**64.1%**	24.3	**68.1%**
cnn	96.2	**50.9%**	65.1	**66.8%**	59.3	**69.7%**

Table 9: Relative improvement, with respect to HTTP/1.1-default, provided by the No-reconf. and Client-reconf. based schemes. Download latency in seconds.

Note that transport and link layers optimizations typically provide 5-10% additional performance improvements, which is still significant for web downloads over WWAN links.

In Table 9 we compare the total benefits of the No-reconf. and the Client-reconf. solutions. We distinguish between two different No-reconf. solutions: No reconf.-I uses HTTP/1.1-Opt while No reconf.-II uses HTTP/1.1-Pipelining. Note that these two techniques are interchangeable since they have the same goal with somewhat similar effect. In both these solutions we also apply all the other No. reconf. solutions at the different layers. The Client-reconf. solution includes Full compression, Delta encoding and extended caching, Parse-and-push, and UDP-GPRS. As we would expect, the Client-reconf. solution leverages extra functionality between the client and the server-side proxy to achieve better performance than the No-reconf. techniques. Additionally we can observe that between the two No-reconf. solutions, the pipelining-based technique achieves better performance than the HTTP/1.1-Opt based technique. This indicates that pipelining is a very crucial technique to improve performance and should be enabled by all web servers.

5. DISCUSSION

The results presented in this paper indicate that the optimizations at all layers of protocol stacks are necessary to achieve significant performance benefits in GPRS environments. The summary presented in Section 4.5 also quantifies the specific benefits of these optimizations at these different layers.

In this section we discuss the following related questions:

- Are these optimization-based benefits specific to 2.5G GPRS-based WWAN networks or do we expect similar performance benefits in the next generation (3G) networks as well? Hence we first present a preliminary case study of a specific 3G CDMA-based network and demonstrate that our observations in this paper would largely extend to these environments.

- Are WWAN environments a special case of low-bandwidth high-latency networks? In particular should we expect that the optimizations studied in this paper in the context of WWANs would also lead to equivalent performance improvements in wired dial-up environments? Hence we next present a similar study for web applications run over wired dialup environments to demonstrate that WWAN environments have significantly different characteristics and performance.

We present these results in two tables (Tables 10 and 11) for FTP and web throughputs respectively as we discuss later.

Finally we will conclude this section with a discussion of limitations of proxy-based solutions and the implications of such limitations in the context of the results presented.

Implications in 3G Networks. To evaluate the potential impact of our results on 3G WWAN environments, we conducted experiments over a commercial CDMA 3G-1X network. In these experiments we used a Samsung VGA1000 CDMA 3G-1X handset (with a maximum downlink datarate of 144Kbps) to measure FTP and web throughputs. The goal of our study was to examine the potential benefits of optimizations to 3G networks. Hence we present the FTP throughputs in Table 10 for different file sizes and

File Size	WWAN 2.5G		WWAN 3G		Wired (dial-up)	
(KB)	T'put	% dgr.	T'put	% dgr.	T'put	% dgr.
1	13.2	**-67%**	16.3	**-87%**	29.1	**-48%**
5	18.1	**-66%**	23.3	**-84%**	35.6	**-37%**
10	18.8	**-54%**	37.8	**-74%**	44.3	**-22%**
50	30.5	**-23%**	87.5	**-39%**	46.1	**-18%**
100	30.5	**-23%**	94.1	**-34%**	45.8	**-19%**

Table 10: Ftp-throughputs in Kbps for different filesizes over different links. Degradation (in %) with respect to the ideal downlink data rates in each case.

the unoptimized web throughput in Table 11 for the four websites and compare them with the corresponding performance of our GPRS experiments. In both these tables we quantify the under-performance of the two data transfer protocols — the under-performance indicates the reduction in actual data throughput in comparison to the maximum achievable downlink data rate in the respective networks. In these tables we can observe that both 2.5G (GPRS) and 3G (CDMA) networks exhibit a significant additional under-performance between FTP throuhgput and web throughput. The FTP throughput in the 3G case for a 100 KB file is about 34% less than the ideal downlink data rate of 144 Kbps, whereas the web download of CNN shows a significantly greater under-performance (i.e., 77% lower than the ideal downlink data rate).

Website	WWAN 2.5G		WWAN 3G		Wired (dial-up)	
(Virtual)	T'put	% dgr.	T'put	% dgr.	T'put	% dgr.
mail	8.5	**-79%**	34.6	**-76%**	37.1	**-34%**
yahoo	13.8	**-66%**	41.2	**-71%**	42.8	**-24%**
amazon	9.6	**-75%**	38.4	**-73%**	43.1	**-23%**
cnn	7.6	**-81%**	32.8	**-77%**	38.5	**-32%**

Table 11: Web download throughputs (in Kbps) over different links without performance optimizations. Performance degradation (in %) is relative to the ideal downlink data rate.

Note that in these experiments we do *not* apply any performance optimizations at any layer. In Table 10 we show the FTP-throughputs achieved over 3G-1X. These results demonstrate that performance mismatch as seen over 2.5G (GPRS) WWAN are also present in the 3G WWAN environments. Hence we expect that our evaluations presented in this paper to extend to these environments as well.

Wired Dial-up Environments. We also experimentally investigated performance of wired (dial-up) environments. We conducted experiments using a standard V90 56Kbps dial-up modem. It is interesting to note that there exists **no** significant mismatch between FTP and web throughputs in the wired dial-up scenarios. The performance degradation (with respect to the ideal data rate of 56 Kbps) of a 100 KB FTP is 19%, while that of a large website like CNN is 32%. This is very unlike the characteristics experienced in WWAN environments. We attribute this difference to the following two reasons. First, the RTTs for dial-up modems (in the 100-150 ms range for 64 byte packets) is relatively lower in comparison to the significantly higher values encountered in WWAN links (in the 600 ms to few seconds range for same packet size). Hence the 'stop-and-go' behavior of default the HTTP protocols in default settings leads to greater under-utilization in WWAN links than it does in dial-up wired environments. Clearly, the optimizations studied in this paper will lead to improvement in download performance over dial-up modems as well [17], but the impact of these techniques are significantly higher in WWANs. Second, the RTT variability on dial-up links is marginal when compared to that of WWAN links. Finally, losses encountered in wired dial-up links are few and far in between (unlike WWAN environments). Hence although the wired dial-up environments have low bandwidths, their impact of data transfer applications is relatively benign. It allows modems to implement additional stateful packet and connection compression techniques that are more difficult to implement in WWAN environments.

Trade-offs in Proxy Deployments over WWANs. Our results demonstrate that proxy-based solutions provides significant benefits to the end-user experience over WWANs. However, the presence of the proxy can split the "end-to-end" properties of an application and has security and other related implications. For example, if the webpage contents are digitally signed by the content provider, then any updates e.g. URL-rewriting or content compression, will violate the security guarantees of the client.

Part of the problem would be solved if the content source itself implemented such proxy functionality. Even more realistically, it may be necessary to provide the clients with the appropriate choice to trade-off performance against end-to-end guarantees. For example, a client browsing news at CNN may accept some susceptibility to insecure data at the cost of significant performance improvements in download latencies. The same client may not use a proxy-based solution when downloading stock-quotes and instead prefer the rigid security properties of end-to-end encryption.

The explicit proxy based solutions define customized mechanisms to provide the best-known benefits to WWAN clients. However, client re-configuration or a client-side software update is an intrinsic requirement of such schemes. This increases the deployment overhead of such schemes higher than the other class of schemes. In many cases it is expensive for cellular operators to provide such updates to existing client equipment. The price performance trade-off of the wireless cellular operators will finally determine the deployment of such mechanisms in WWAN environments.

6. RELATED WORK

Researchers have examined various optimization choices at the different layers of the protocol stack in the context of both wired, wireless, and also for WWAN environments. However, much of the prior research has focussed mainly on isolated performance optimizations.

Piror research in WWANs have primarily focussed on the following three aspects: (a) improving TCP performance over WWANs (e.g. [14, 33]), (b) passive analysis of TCP traffic traces (e.g. [8]), and, (c) cross-layer interaction and optimizations of TCP with the link-layer [25, 30]. Our work in this paper differs from all prior work in WWANs in several ways. In our study (1) we quantify the causes of poor *application performance* and examine the user experience over WWANs, (2) we measure the different components that contribute to the latencies during web downloads for a range of popular websites, (3) we use *virtual web hosting* as an important construct to perform repeatable and reproducible web browsing experiments over WWANs, (4) we benchmark all standard web browsers, protocols, and techniques with respect to their performance, and, (5) we implement and study a wide selection of optimization techniques at different layers and their cross layer interactions on application performance.

At the link layer, TULIP [11] describes a transport unaware ARQ mechanism to improve TCP performance. A. Chokalingam et al. in [15] and Ayanoglu et al. in [2] examine the interaction between ARQ and FEC-based link recovery mechanisms in the context of wireless networks through detailed simulations and is related to our discussion in Section 4.4. Balakrishnan et. al. [4] use explicit loss notification (ELN) from the wireless link, mostly in the context of wireless LANs, to improve the performance of applications like HTTP.

For the transport layer new mechanisms have been defined to improve the end-to-end performance of reliable applications in wireless LAN environments. Examples of this are Snoop [5], I-TCP [3], M-TCP [34] etc. Unfortunately, solution meant for wireless LANs may not work that well over WWAN. Researchers have also demonstrated that the GPRS link layer and TCP do not adversely interact with each other [25, 31]. Similarly, link layer (RLP) retransmissions for 3G-1x links ensure packet loss probability of less than 1% that minimizes impact on TCP [14]. While our results agree with these above observations that the link-layer is generally well-tuned for transport TCP protocol to operate over WWANs, the end-result (*i.e.* user experience) for TCP-based applications like web browsing remains remarkably different. Our work also demonstrates that employing appropriate optimizations at the application and session layers (as described in this paper) can provide significant benefits to actual user (web) experience.

In other important works, R. Ludwig et. al. [31] examine the performance of TCP over GSM cellular links and subsequently propose specific link layer mechanisms (e.g. frame size adaptations) that are necessary to improve TCP performance in such environments [30]. In [33], P. Sinha et al. introduce WTCP as one of the first solution to overcome performance problems seen for TCP over CDPD-based WWAN links. Many of the link-related performance issues (high RTTs, black-outs etc.) observed in CDPD-based WWANs are similar to that observed in GPRS and 3G-1x based WWANs. In [14], M. C. Chan and R. Ramjee proposed *ACK Regulator* for improving TCP performance over CDMA-based 3G-1X links. Unlike GSM-based GPRS, 3G-1X links exhibit much higher delay and rate variations and transport-layer optimizations like *ACK regulator* can be used to significantly benefit TCP performance in such environments.

IETF RFC 3135 [16] and RFC 3481 [18] provide further details of different mechanisms to improve TCP performance in different wireless environments. Research has also investigated some TCP performance issues over GPRS, e.g., a large-scale passive analysis of end-to-end TCP flows [8].

Proxy and caching based schemes to improve web performance, similar to the ones we have explored in this paper for WWAN environments, have been already extensively explored in the prior literature mostly in the context of wired environments, e.g. Cache Digests [23], response-aliasing in web transactions [21] etc. A detailed description of caching schemes is presented in [7]. Similarly, delta encoding is also a well-known and useful technique to improve HTTP performance [26]. WebExpress [19] from IBM defines some application level techniques, including caching, differencing, and header-reduction mechanisms, and is related to some of the techniques explore in this paper. In the context of WWAN environments, Liljeberg et. al. developed Mowgli Communication Architecture [24] that uses a pair of proxies to define employs a custom protocol tailored for the GSM link. Their solution is similar to (UDP-GPRS) [12].

The Wireless Application Protocol [1] is another related mechanism that employs explicit proxies (optional but highly recommended) to improve web experience of WWAN users. The optimization choices discussed in this paper can be applied in the context of WAP 2.0, specifically for improvements suggested in "wireless-profiled" TCP and HTTP [1].

7. CONCLUSIONS

We conducted a detailed comparative performance study of a wide selection of optimizations choices applicable for WWANs. Ours is the first significant study to have attempted to address important questions like: Why is the web so slow over WWANs? Even though TCP is relatively well-tuned to perform efficiently in these environments, why is the performance of HTTP applications significantly worse? While prior studies have examined the problems of TCP in WWAN environments, we are not aware of any prior research that presents a detailed evaluation of application performance. Our performance study also provides important insights in understanding *what* optimization choices can yield *how much* benefit. The performance optimizations at each individual layer, studied in this paper, leverage well-adapted and optimized lower layers. This was done to avoid any inefficient cross-layer design including adverse inter-layer interactions.

The following are some of our important observations:

1. There is a significant mismatch in the performance of default HTTP protocols and its underlying transport mechanism TCP. Unlike the wired (e.g. dial-up) environments, we find that standard web browsers are unable to exploit the meagre resources of the WWAN links.The achieved throughput is sometimes 70% lower than the ideal downlink data rate (Table 11).

2. Significant application performance benefits can be realized (about 48-61% improvements) by suitable optimizations implemented at the application and session layers (Figure 8).

3. Proxy-based solutions are most effective in improving in application performance than non-proxy based approaches (Table 9).

Our performance study has broad implications. The significant benefits resulting from using HTTP pipelining in WWANs highlights the need to implement this feature in all commercial class web servers and standard web browsers. Hence, appropriate support from web server vendors, content providers and browser designers will go a long way in the success of the next generation 'mobile' Internet.

We find that a collective suite of performance optimizations implemented using proxies at different layers can reduce the client-response times by at least a factor of two. This is possible because the proxies are specifically aware of the characteristics of the WWAN environment and hence makes more 'intelligent' decisions to adapt the performance of data delivery mechanisms. Such awareness of client characteristics is crucial for improving the overall end user-experience. This can imply any one of the following two things: (1) Proxy-based solutions should not be restrictively viewed as a short-term solution. Instead, cellular operators should design, implement and deploy such proxies within their network and end-users should be given the choice to use such proxies, thus, trading off security with performance. Such an approach may be acceptable in certain scenarios. (2) The intelligence of the proxies should be implemented in the web servers, content providers, as well as the web browsers. Such an approach will maintain "end-to-end"ness of the protocols, however, will require significant collaborative effort between all these diverse vendors of different applications.

Other than performance, it is important to consider the trade-offs between the cost and the ease of deployment associated with such proxy installations. As previously discussed in our study, transparent proxies are the easiest to deploy since they require no changes or configuration to the mobile-end client systems. However, from a performance perspective, dual-proxy based solutions seems to provide the most significant benefits. Unfortunately, such an approach requires either a reconfiguration or a software update in the mobile client. This increases its deployment overhead. In many cases it is expensive for the cellular operators to provide such updates to the existing client equipment.

This paper presents an extensive study of the performance of web applications over GPRS-based WWAN environments. In Section 5 we also present our preliminary work in understanding the performance characteristics in other WWAN environments, e.g. 3G technologies like CDMA 2000. We believe that a further detailed characterization of these environments will be very useful. Our hope is that others will also perform similar studies of *actual user experience* over such wireless environments (e.g. W-CDMA UMTS and CDMA 2000) so that extensive benchmarks could be obtained and eventually lead to adoption of a "best of both worlds" solution.

8. REFERENCES

[1] http://www.wapforum.org.

[2] E. Ayanolgu, S. Paul, T. Porta, K. Sabnani, and R. Gitlin. Airmail: A link-layer protocol for wireless networks. *ACM Wireless Networks*, 1(1), 1995.

[3] A. Bakre and B.R. Badrinath. I-tcp: Indirect tcp for mobile hosts. In *Proc. of IEEE ICDCS*, 1995.

[4] H. Balakrishnan and R.H. Katz. Explicit loss notification and wireless web performance. In *Proc. of IEEE Globecom*, 1998.

[5] H. Balakrishnan, R.H. Katz, and S. Seshan. Improving tcp/ip performance over wireless networks. In *ACM Mobicom*, 1995.

[6] P. Barford and M. Crovella. A performance evaluation of hyper text transfer protocols. In *ACM Sigmetrics*, 1999.

[7] G. Barish and K. Obraczka. World wide web caching: Trends and techniques. *IEEE Communications Magazine*, May 2000.

[8] P. Benko, G. Malicsko, and A. Veres. A large-scale, passive analysis of end-to-end tcp performance over gprs. In *Proc. of IEEE INFOCOM*, 2004.

[9] G. Brasche and Walke B. Concepts, services and protocols of the new gsm phase 2+ general packet radio service. *IEEE Communications Magazine*, August 1997.

[10] C. Bettssetter, H. Vogel, J. Eberspacher. Gsm phase 2+ general packet radio service gprs: Architecture, protocols, and air interface. *IEEE Communication surveys*, 2(3), 1999.

[11] C. Parsa and JJ Garcia-Luna-Aceves. Tulip: A link-level protocol for improving tcp over wireless links. In *Proc. of IEEE WCNC*, 1999.

[12] R. Chakravorty, A. Clark, and I. Pratt. Gprsweb: Optimizing the web for gprs links. In *ACM Mobisys*, 2003.

[13] R. Chakravorty, S. Katti, I. Pratt, and J. Crowcroft. Flow aggregation for enhanced tcp over wide-area wireless. In *Proc. of IEEE INFOCOM*, 2003.

[14] M.C. Chan and R. Ramjee. Tcp/ip performance over 3g wireless links with rate and delay variation. In *ACM Mobicom*, 2002.

[15] A. Chockalingam and M. Zorzi. Wireless tcp performance with link layer fec/arq. In *Proc. of IEEE ICC*, 1999.

[16] S. Dawkins, G. Montonegro, M. Kojo, V. Magret, and N. Vaidya. End-to-end performance implications of links with errors. In *RFC 3155, IETF*, 2001.

[17] H.F. Neilsen et. al. Network performance effects of http/1.1, css1, and png. In *ACM Sigcomm*, 1997.

[18] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov and F. Khafizov. Tcp over second (2.5g) and third (3g) generation wireless networks. In *RFC 3481, IETF*, 2003.

[19] B.C. Housel and D.B. Lindquist. Webexpress: A system for optimizing web browsing in a wireless environment. In *ACM Mobicom*, 1996.

[20] R. Kalden, I. Meirick, and M. Meyer. Wireless internet access based on gprs. In *IEEE Personal Communications*, 2000.

[21] T. Kelly and J. Mogul. Aliasing on the world wide web: Prevalence and performance implications. In *World Wide Web (WWW) Conference*, 2002.

[22] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[23] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3), 2000.

[24] M. Liljeberg and T. Alanko and M. Kojo and H Laamanen and K. Raatikainen. Optimizing world-wide web for weakly-connected mobile workstations: An indirect approach. In *International Workshop on Services in Distributed and Networked Environments (SDNE)*, 1995.

[25] M. Meyer. Tcp performance over gprs. In *Proc. of IEEE WCNC*, 1999.

[26] J.C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for http. In *ACM Sigcomm*, 1997.

[27] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and implications. In *ACM Sigcomm*, 2000.

[28] V.N. Padmanabhan and J.C. Mogul. Improving http latency. *Computer Networks and ISDN Systems*, 28(1), December 1995.

[29] R. Fielding et al. Hypertext transfer protocol – http/1.1. In *RFC 2616, IETF*, 1999.

[30] R. Ludwig, A. Konrad and A. Joseph. Optimizing the end-to-end performance of reliable flows over wireless links. In *ACM Mobicom*, 1999.

[31] R. Ludwig, B. Rathonyi, A. Konrad, K. Oden and A. Joseph. Multi-layer tracing of tcp over a reliable wireless link. In *ACM Sigmetrics*, 1999.

[32] P. Rodriguez, S. Mukherjee, and S. Rangarajan. Session-level techniques to improve web-browsing performance over wide-area wireless links. Bell Labs

Technical Report, May 2003.

[33] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Wtcp: A reliable transport protocol for wireless wide-area networks. In *ACM Mobicom*, 1999.

[34] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of tcp over mobile internetworks (mtcp). In *Workshop on Mobile Computing Systems and Applications*, 1994.