GPRSWeb: Optimizing the Web for GPRS Links

Rajiv Chakravorty, Andrew Clark and Ian Pratt
E-mail: {firstname.lastname}@cl.cam.ac.uk
University of Cambridge Computer Laboratory,
JJ Thomson Avenue, Cambridge CB3 0FD, U.K.

ABSTRACT

The General Packet Radio Service (GPRS) is being deployed by GSM network operators world-wide, and promises to offer users "always-on" data access at bandwidths comparable to that of conventional fixed-line telephone modems. Unfortunately, many users have found the reality to be rather different, experiencing very disappointing performance when, for example, browsing the web over GPRS.

In this paper we investigate what causes the HTTP protocol and its underlying transport TCP to underperform in a GPRS environment. We examine why certain GPRS network characteristics interact badly with TCP to yield problems such as: link under-utilization for short-lived flows, excess queueing for long-lived flows, ACK compression, poor loss recovery, and gross unfairness between competing flows. We also show that many web browsers tend to be overly aggressive, and by opening too many simultaneous TCP connections can aggravate matters.

We present the design and implementation of GPRSWeb – a mobile HTTP proxy system that mitigates many of the performance problems with a simple software update to a GPRS mobile device. The update is a 'client proxy' that sits in the mobile device, and communicates with a 'server proxy' located at the other end of the GPRS link close to the wired-wireless border. The dual proxy architecture collectively implements a number of key enhancements – an aggressive caching scheme that employs content-based hash keying to improve hit rates for dynamic content, a preemptive push of web page support resources to mobile clients, resource adaptation to suit client capabilities, delta encoded data transfers, DNS lookup migration, and a UDP-based reliable transport protocol that is specifically optimized for use over GPRS. We show that these enhancements result in significant improvement in overall WWW performance over GPRS.

1. INTRODUCTION

World over, GSM cellular networks are being upgraded to support the General Packet Radio Service (GPRS). GPRS offers an "always on" connectivity to mobile users, with wide-area coverage and data rates comparable to that of conventional fixed-line telephone modems. This holds the promise of making ubiquitous mobile access to IP-based applications and services a reality.

However, despite the momentum behind GPRS, surprisingly little has been done to evaluate WWW performance over GPRS. There are some interesting simulation studies [1, 4] on TCP performance, but we have found actual deployed network performance to be somewhat different.

Some of the web performance issues observed with GPRS are shared, to some extent, with wireless LANs like 802.11b (*WiFi*), satellite systems, and other wide-area wireless schemes such as Metricom Ricochet and Cellular Digital Packet Data (CDPD). However, we feel that GPRS presents a particularly challenging environment for achieving good application (web) performance.

Past research has investigated TCP (and also HTTP) performance over a number of wide-area wireless links such as Ardis, Metricom Richochet, CDPD and GSM. However, the real inhibitors to a better web browsing experience are typically related to the underlying network characteristics, which as we shall see, are somewhat different for GPRS.

In this paper we set out to explore questions like:

1. What are the "typical" GPRS network characteristics?
2. What are the practical performance problems using TCP and HTTP over GPRS?
3. What overall benefit can we achieve using various application level optimization schemes over GPRS?

In this paper, we present our practical experiences over production GPRS networks, and our attempts to build a system that optimizes WWW performance over GPRS. After a brief overview on GPRS in the next section, we summarize our work to characterize GPRS link behaviour in section 3. Section 4 identifies particular problems experienced by TCP flows over GPRS, and section 5 examines how these are exacerbated by application-layer protocols such as HTTP.

In section 6, we present the design and implementation of our GPRSWeb proxy system – a dual-proxy system architecture consisting of a 'client proxy' and a 'server proxy'. While the GPRSWeb client proxy resides in the mobile device, the GPRSWeb server proxy is located in the network in close vicinity to the wired-wireless border. GPRSWeb aims to improve WWW performance with an optimized transport protocol specifically tailored for GPRS, an advanced caching scheme, server controlled *parse-and-push* functionality, data compression, and document delta encoding. GPRSWeb requires no instrumentation or modifications to be made to either web browsers or servers.

Section 7 discusses our experimental test bed setup and in section 8 we present an evaluation of the performance our proxy system. We conclude with a discussion of related work and summarize our experience with the system.

2. THE GPRS OVERVIEW

GPRS is a bearer service for GSM - a wireless extension to packet data networks. Two new nodes (see, figure 6) have been added to the traditional GSM network to support GPRS: the SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Suport Node). The SGSN node acts as a packet switch that performs signaling similar to a mobile switching center (MSC) in GSM, along with cell selection, routing and handovers between different Base Switching Centers (BSCs). It also controls the mobile terminal'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.

A mobile terminal (MT) wishing to use GPRS will first *attach* itself to the network through a signaling procedure. The *attach* procedure can be performed either when the MT is switched on or when the user wishes to transfer packet data. Depending upon the type of GPRS device class, it can connect either to circuit switched or to packet switched services, or both simultaneously [1]. Mobile terminals are classified according to the number of time slots they are capable of operating on simultaneously. For example, many current GPRS devices are classified as "3+1" meaning that at any given time they can listen to 3 downlink channels (from base station to mobile), but can only transmit on 1 uplink channel to the base station.

GPRS copes with a wide range of radio conditions by making use of 4 different coding schemes (CS-1 TO CS-4) [1][5] with varying levels of FEC (forward error correction). Most of the currently deployed GPRS networks support only CS-1 and CS-2 [34] – the other two are not used as error rates would be typically too high to be useful. CS-4 removes FEC correcting capabilities altogether. The effective GPRS data rate is slightly less, due to protocol header overhead and signalling messages. The RLC (radio link control) layer attempts to provide reliable in-order delivery of packets.

Radio resources (TDMA time slots) of a cell are shared between all GPRS and GSM mobile stations located within a cell. Most network operators typically configure the network to give GSM (voice) calls strict priority over GPRS for time slot allocation. The time slots available for GPRS use, known as packet data channels (PDCHs), are then dynamically allocated (using the *capacity on demand* principle [5]) between mobile terminals with data to send or receive. GPRS can multiplex time slots between different users, and can also allow multiple time slots to be used in parallel to increase bandwidth to/from a particular mobile terminal.

The latest GPRS Release (1999) defines several QoS parameters to meet the application requirements for different levels of network QoS. However, currently deployed networks typically only support a single *best-effort* service class [34]. Further information about GRPS network design and operation can be found in [1-5].

3. GPRS LINK CHARACTERIZATION

GPRS [1, 5], like other wide-area wireless networks, exhibits many of the following characteristics: low and fluctuating bandwidth, high and variable latency, and occasional link 'blackouts' [3, 7]. To gain clear insight into the characteristics of the GPRS link, we have conducted a series of link characterization experiments. The-se have been repeated under a wide range of conditions, using different models and manufacturer of handsets, and different network operators located in several European countries. We have found no major performance differences between the network operators, and variation between different handsets of similar GPRS device class is minimal. Details of how these tests were conducted (uplink and downlink latency measurements, tools etc.) can be found in [2]. [33] also gives a comprehensive description on GPRS link characterization in the form of a separate technical report. Below, we enunciate some key findings:

High and Variable Latency:- GPRS link latency is very high, 600ms-3000ms for the downlink and 400ms-1300ms on the uplink. Round-trip latencies are 1000ms or more. The delay distribution is shown in figure 1. The link also has a strong tendency to 'bunch' packets; the first packet in a burst is likely to be delayed and experience more jitter than following packets. This indicates that a substantial proportion of the latency is incurred when the link to a mobile terminal transitions from previously being idle [2]. Packets that are already queued for transmission can then follow the first out over the radio link without incurring additional jitter. The additional latency for the first packet is also typically due to allocation time of the temporary block flows (TBFs). Since most current GPRS terminals allocate and release TBFs immediately (implementation based on GPRS 1997 release), applications (such as TCP) that can transfer temporally-separated data (and ack) packets may end up creating many small TBFs that can each add some delay (approx. 100-200msec) during data transfer.

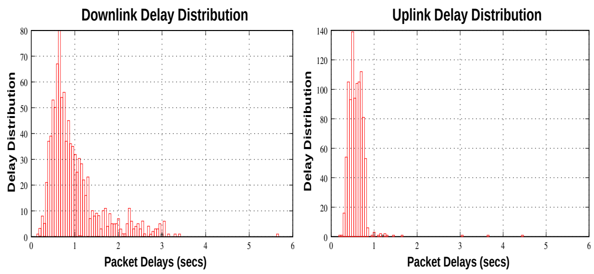


Figure 1: Single packet time-in-flight delay distribution plots showing (a) downlink delay (b) uplink delay distribution. Measurements involved transfer of 1000 packets with random intervals > 4s between successive packet transfers.

Fluctuating Bandwidth:- We observe that signal qu-ality leads to significant (often sudden) variations in perceivable bandwidth by the receiver. Sudden signal quality fluctuations (good or bad) commensurately impacts GPRS link performance. Using a "3+1" GPRS phone such as the Ericsson T39 (3 downlink channels, 1 uplink), we observed a maximum raw downlink throughput of about 4.15 KB/s (33.2 Kb/s), and an uplink throughput of 1.4 KB/s (11.2 Kb/s). Using a "4+1" phone, the Motorola T280, we measured an improved maximum bandwidth of 5.5 KB/s (44 Kb/s) in the downlink direction. If CS-2 coding scheme was used, then using a '3+1' and "4+1" phone, we could achieve a theoritical maximum of 40.2 Kb/s and 53.6 Kb/s respectively. Some other factors contribute to throughput values lower than the maximum possible, see [2, 4].

Packet Loss:- The radio link control (RLC) layer in GPRS uses an automatic repeat request (ARQ) scheme that works aggressively to recover from link layer losses. Thus, higher-level protocols (like IP) rarely experience non-congestive losses. Packets can be lost over the GPRS link during, (1) deep fading leading to bursty losses, and (2) cell re-selections due to the cell (or routing area) update procedure resulting in link a 'black-out' condition. In both cases, consecutive packets in a window are usually lost.

Link Outages:- Link outages are common while moving at speed, or when passing through tunnels or other radio obstructions. Nevertheless, we have also noticed outages during stationary conditions. The observed outage interval will typically vary between 5 and 40s. Sudden signal quality degradation, prolonged fades and intra-zone handovers can lead to such link blackouts. When link outages are of short duration, packets are simply delayed and are lost in few cases. In contrast, when outages are of higher duration there tend to be burst losses. We have also observed specific cases of link resets, where a mobile terminal would stall and stop listening to its temporary block flow (TBF). In such cases, we had to terminate and restart the point-to-point (PPP) session.

4. TCP PERFORMANCE OVER GPRS

In this section we discuss TCP performance problems over GPRS. In particular, we concentrate on connections where the majority of data is being shipped in the downlink direction, as this corresponds to the prevalent behaviour of most mobile applications, such as web browsing, file download, reading email, news etc. A more comprehensive description on TCP performance problems over GPRS can be found in [3].

TCP Start-up Performance:- Figure 2(a) shows a close up of the first few seconds of a connection, displayed alongside another connection under slightly worse radio conditions. An estimate of the link bandwidth delay product (BDP) is also marked, approximately 10KB[1]. For a TCP connection to fully utilize the available link bandwidth, its congestion window must be equal or exceed the BDP of the link. We can observe that in the case of good radio conditions, it takes over 7 seconds to ramp the congestion window up to a value of link BDP from when the initial connection request (TCP's SYN) was made. Hence, for transfers shorter than about 18KB, TCP fails to exploit the meagre bandwidth that GPRS makes available to it. Since many HTTP objects are smaller than this size, the effect on web browsing performance can be dire.

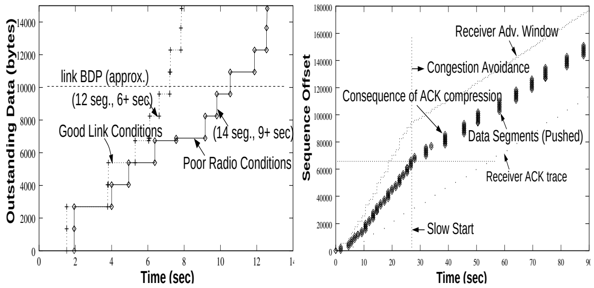


Figure 2: Plot (a) shows that slow-start takes over 7 seconds to expand the congestion window sufficiently to enable the connection to utilise the full link bandwidth. (b) shows the characteristic exponential congestion window growth due to slow-start (SS).

ACK Compression:- A further point to note in figure 2(b) is that the sender releases packets in bursts in response to groups of four ACKs arriving in quick succession. Receiver-side traces show that the ACKs are generated in a smooth fashion in response to arriving packets. The 'bunching' on the uplink is due to the GPRS link layer. This effect is not uncommon, and appears to be an unfortunate interaction that can occur when the mobile terminal has data to send and receive concurrently. ACK bunching or compression not only skews upwards the TCP's RTO measurement but also affects its self-clocking strategy. Sender side packet bursts can further impair RTT measurements.

Excess Queuing:- Due to its low bandwidth, the GPRS link is almost always the bottleneck of any TCP connection, hence packets destined for the downlink get queued at the gateway onto the wireless network (known as the GGSN node in GPRS terminology, see figure 9). However, we found that the existing GPRS infrastructure offers substantial buffering: UDP burst tests indicate that over 120KB of buffering is available in the downlink direction. For long-lived sessions, TCP's congestion control algorithm could fill the entire router buffer before incurring packet loss and reducing its window. Typically, however, the window is not allowed to become quite so excessive due to the receiver's flow control window, which in most TCP implementations is limited to 64KB unless *window scaling* is explicitly enabled. Even so, this still amounts to several times the BDP of unnecessary buffering, leading to grossly inflated RTTs due to queueing delay. Figure 3 (b) shows a TCP connection in such a state, where there is 40KB of outstanding data leading to a measured RTT of tens of seconds. Excess queueing exacerbates other issues:

[1] The estimate is approximately correct under both good and bad radio conditions, as although the link bandwidth drops under poor conditions the RTT tends to rise.

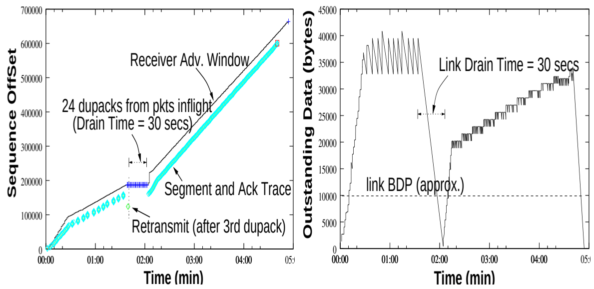


Figure 3: Case of timeout due to a dupack(sack). Plot (a) shows the sender sequence trace and (b) shows corresponding outstanding data.

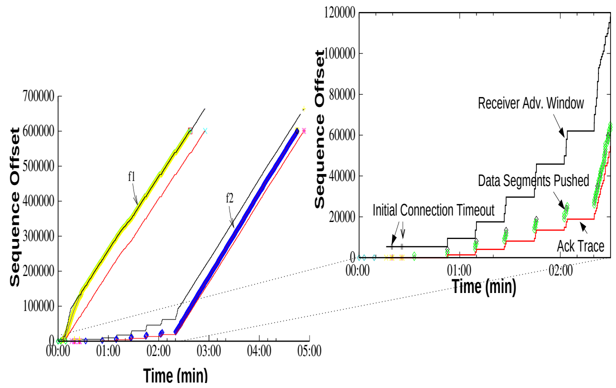


Figure 4: Close-up of time sequence plots for two concurrent file transfers over GPRS, where `f2` was initiated 10 seconds after `f1`.

- **Inflated Retransmit Timer Value.** RTT inflation results in an inflated retransmit timer value that impacts TCP performance, for instance, in cases of multiple loss of the same packet [6].

- **SYN timeout.** Excess queuing caused by long-lived flows often results in attempts to establish new connections timing-out before completing the 3-way handshake. [6].

- **Problems of Leftover (Stale) Data.** For downlink channels, the queued data may become obsolete when a user aborts a web download and abnormally terminates the connection. Draining leftover data from such a link may take many seconds.

- **Higher Recovery Time.** Recovery from timeouts due to dupacks (or sacks) or coarse timeouts in TCP over a saturated GPRS link takes many seconds. This is depicted in figure 3(a) where the *drain time* is about 30s.

TCP loss recovery over GPRS:- Figure 3(a)-(b) depicts TCP's performance during recovery due to reception of a dupack (in this case a SACK). The point to note here is the very long time it takes TCP to recover from the loss, on account of the excess quantity of outstanding data. Fortunately, use of SACKs ensures that packets transferred during the recovery period are not discarded, and the effect on throughput is minimal. This emphasises the importance of SACKs in the GPRS environment. In this particular instance, the link condition happened to improve significantly just after the packet loss, resulting in higher available bandwidth during the recovery phase.

Fairness between flows:- Excess queueing can lead to gross unfairness between competing flows. Figure 4 shows a file transfer (f2) initiated 10 seconds after transfer (f1). When TCP transfer (f2) is initiated, it struggles to get going. In fact it times out twice on initial connection setup (SYN) before being able to send data. Even after establishing the connection, the few initial data packets of f2 are queued at the CGSN node behind a large number of f1 packets. As a result, packets of f2 perceive very high RTTs (16-20 seconds) and bear the full brunt of excess queueing delays due to f1. Flow f2 continues to badly underperform until f1 terminates. Flow fairness turns out to be an important issue for web browsing performance, since most browsers open multiple concurrent HTTP connections [10]. The implicit favouring of long-lived flows often has the effect of delaying the "important" objects that the browser needs to be able to start displaying the partially downloaded page, leading to decreased user perception of performance.

5. WWW PERFORMANCE OVER GPRS

The results from the preceding section show that TCP performs poorly in a wide-area wireless GPRS environment. In this section, we briefly review the key issues related to browser, and specifically, web performance over GPRS. Further information related to web performance issues over GPRS can be found in [7].

Figure 5 shows a sample connection setup overhead for a web connection. Typically, the setup overhead in a web connection could entail two round-trips. In the first round-trip, a web client resolves the requested Uniform Resource Identifier (URI) with a check to a local Domain Name Server (DNS) cache for an entry to the requested URL. If it is present in the local DNS cache and has not yet timed-out, the cache entry is directly used to prevent an unnecessary DNS server lookup. However, if the local cache search fails, the client then makes a request to the DNS server, which in the GPRS case, is located on the other end of the wireless link. As RTTs over GPRS are high, a significant DNS lookup overhead (higher than a single GPRS RTT) is entailed in the process.

In the second round-trip, however, the web client in a mobile device initiates a TCP connection with the remote server. As again in this case, every TCP connection will have to proceed through a 3-way TCP handshake, which means an ad-

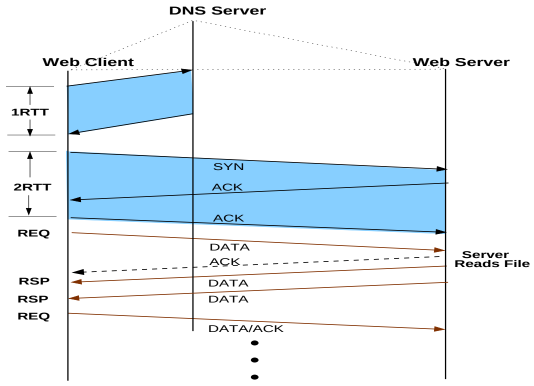


Figure 5: Web Connection SetUp Overhead

ditional RTT is incurred before the connection is used. Once this is done, the client can proceed making a request to the server followed by normal data transfer from the server. Thus the total delay for the first TCP connection to successfully initiate the first HTTP request can be as high as 2 GPRS round trips.

Browser behaviour is obviously crucial to a web performance over any given network. Unfortunately, most current web browsers are tuned for LAN environments and often perform poorly in a resource restricted setting. In [7], experiments conducted show that web browsers (e.g. mozilla) tend to open multiple concurrent TCP connections over a link simultaneously. The inherent nature of TCP's congestion control algorithm implies that N connections will be N times more aggressive when compared to a single TCP connection when sharing a bottleneck link. Typically, web browsers that open a number of concurrent TCP connections do so to grab a greater share of the link bandwidth. Also, with more connections, browsers implicitly avoid *head-of-line* (HOL) [14] blocking problems. An aggressive browser will obviously reap benefits over conventional high bandwidth links shared by many users.

Using multiple connections over 'long-thin' GPRS links can be deleterious: First, protocol control (SYNs/ACKs-/FINs) overhead associated with higher numbers of connections is high. This is further exacerbated by the overhead of the protocol headers (i.e. TCP+IP+SNDCP+L-LC=55 bytes, as in [34]) even for data packets that are exchanged over the link. Secondly, the 3-way handshake delay while establishing a TCP connection can be significant due to the high latency of GPRS links. Further, it can take only a few RTTs for multiple concurrent connections to exceed the GPRS CGSN router downlink bandwidth-delay product (BDP) value. The exponential nature of the slow-start phase combined with packets from multiple flows can quickly lead to excess queuing over the downlink. As a result, any subsequent new TCP connection will have a high chance of timing out during its initial connection request phase. New connections will endure high RTTs, causing them to severely underperform, with an additional probability of spurious timeouts. Experiments from [7] over production GPRS networks show that aggressive web browsers tend to saturate the downlink GPRS GGSN buffers. This has been shown to negatively impact web performance over GPRS.

Many of the widely deployed web browsers continue to use non-persistent connections (HTTP/1.0), employing a new TCP connection for every object downloaded. Use of HTTP/-1.1 persistent, connections, where the browser and server employ the same TCP connection for transfer of multiple objects are gradually becoming more widespr-ead, and help reduce connection setup delay and overhead. However, the full benefits of HTTP/1.1 can not be realised without making use of *pipelining*, where multiple outstanding requests are permitted on the same connection. Without pipelining, a RTT delay is incurred between objects on the same connection, and worse, slow-start must be performed at the start of every object since the connection will have gone idle. Unfortunately, use of pipelining is currently almost non existent. Experiments in [7] show that HTTP request pipelining can improve web performance over GPRS.

6. THE GPRSWEB PROXY MODEL

Having identified the causes of poor WWW performance in a GPRS environment, we now report on our attempts to overcome them. Clearly, performance could be improved by making modifications to the HTTP and TCP protocols to better suit the GPRS environment. However, any approach that relies on modifications to web servers or web browsers would at best take years to achieve wide-spread deployment. Our approach has been to use the existing HTTP proxy mechanism to enable us to insert a pair of translating proxy servers in to the HTTP request/response stream that together implement a number of techniques to enhance performance. This enables GPRSWeb to be both browser and server independent.

GPRSWeb uses a pair of special proxy servers, located on either side of the GPRS link. Between the proxies, a custom protocol is employed to reduce traffic volume over the GPRS link and attempt to mitigate the effects of GPRS's high RTT. The 'client proxy' must be installed on the mobile device. As part of the installation, the web browser is configured to route all HTTP requests through this proxy via a local TCP connection using the traditional *loopback* interface. As shown in figure 6, the client proxy communicates with a 'server proxy', located on the other end of the GPRS link. The server proxy makes requests to the wired network on behalf of the client, and sends back responses. The server proxy is capable of servicing large numbers of simultaneous mobile clients. The cache content at the server proxy is shared.

The GPRSWeb proxy model implements the following mechanisms to improve performance:

GPRSWeb Protocol. Due to the problems identified earlier, we do not use TCP as the transport protocol between the proxies either side of the GPRS link. Instead, we use a custom transport protocol (which we call GPRSWeb protocol hereafter) that runs over UDP and implements ordered, reliable, message transfer. The protocol is optimised for GPRS link characteristics, and minimises link traversals and responds efficiently in the event of the patterns of packet loss

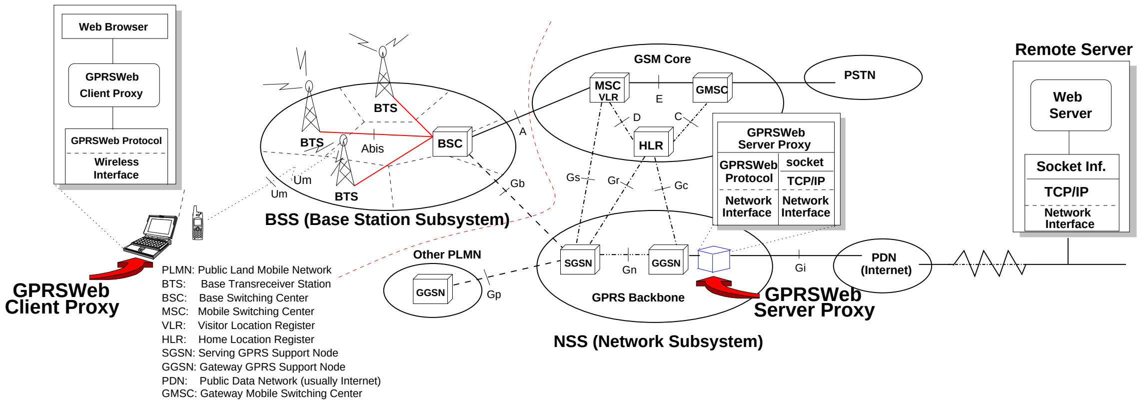


Figure 6: GPRSWeb System Architecture and Components

we commonly observe. It achieves substantially better link utilization than TCP.

Extended Caching. Client-side caching improves performance by eliminating some network round-trips and reducing the amount of data exchanged over the GPRS link. However, traditional browser caches do not yield the full potential benefit due to the nature of the HTTP caching mechanism, and pessimistic cache control directives contained in many web pages.

The GPRSWeb client proxy implements a client-side cache that replaces the browser's persistent (disk) cache. A custom caching protocol is used between the client and server proxies that enables better hit rates by using `SHA-1` fingerprints [29] of objects to determine whether they have actually changed or not. The protocol thus eliminates unnecessary object transfers over the GPRS link, and makes better use of the limited size cache available in the mobile device. The server-side proxy also implements a traditional HTTP cache to reduce bandwidth requirements on the wired network, and thus can take the place of existing proxy caches that are already commonly deployed by ISPs.

Data Compression and Delta Encoding. GPRSWeb also compresss data before sending it over the wireless link, reducing transfer size and thereby improving response time. Data is compressed using *gzip* compression, unless it is already in a compressed format (e.g. JPEG images, Zip Archives). A separate string table is used for HTTP headers, resulting in better compression. When the server-side proxy detects that a previously cached object has been updated, it tries using the VCDiff [31] algorithm to encode the differences between the old and new objects. The compressed *deltas* [15] are sent in place of the new version if they would result in a smaller transfer.

Parse-and-Push Operation. Most web pages contain a number of images and other objects that make up the page structure, e.g. button graphics, spacers, style sheets, frames etc. These objects are requested by a browser after parsing the HTML documents. A round trip delay is normally incurred before transfer of these objects can commence. The *parse-and-push* mechanism in the GPRSWeb server proxy parses HTML objects, and pro-actively starts *pushing* objects towards the GPRSWeb client cache if the link would otherwise be idle.

6.1 GPRSWeb Proxy Design

We have designed and implemented the GPRSWeb proxy architecture by splitting the client and server functionality across a number of components, some of which are shared. Figure 7 show the components used in the client and the server structure. We now discuss the components that constitute the GPRSWeb system design.

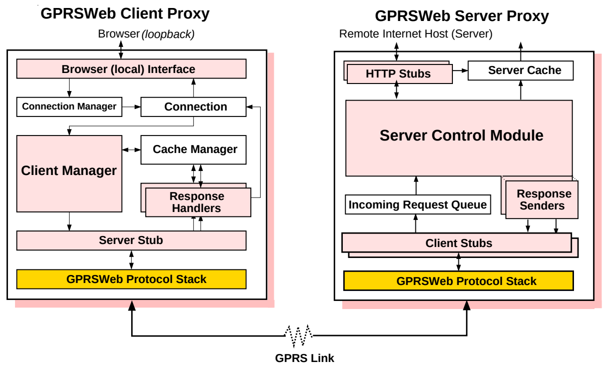


Figure 7: Client and Server Proxy Structure

As shown in the client structure of figure 7, the **Connection Manager** accepts TCP connections from the web browser and passes them to a **Connection** object. This queries the **Client Cache Manager**, and in case of a miss, invokes a **Server Stub** to issue a request to the server proxy. Unique identifiers are assigned to each request made to the server proxy, and are used to invoke a **Response Handler** to process the reply. The response handler also interacts with the cache manager to update cache state as necessary. The object is then returned to the pending browser connection.

In the server proxy, **Client stubs** examine messages received by the protocol stack from a client and takes action depending upon message type. Object request messages are processed by **Server Manager**, which functions very similar to the client manager, but seeks responses from the **Server Cache Manager** and the **HTTP Stubs** if necessary. The HTTP Stubs contact web servers to download or check the freshness of objects. Thus, any DNS lookups required are performed on the server proxy and not over the GPRS link. The client stubs co-ordinate data compression and other optimizations before the response is finally sent back to the client proxy.

6.2 GPRSWeb Protocol

The GPRSWeb transport protocol avoids TCP's connection setup and slow-start costs, and exploits knowledge of GPRS's particular link characteristics to optimise performance.

The basic unit of transfer is the *segment*, each of which is carried in a separate UDP packet. UDP is used to take advantage of the port multiplexing facilities and the UDP checksum. Segments are sequentially numbered, and are of two types: *Normal-priority* and *Low-priority*. Low-priority is typically used for background transfers, e.g. pushing cache updates to a client; normal-priority is used for everything else. The queues are serviced with strict priority. Segments in the low priority queue may be promoted if, for example, the server proxy explicitly receives a request for an object that is currently preemptively being pushed to the client.

Since we expect missing segments to be rare over GPRS (due to the underlying reliable RLC layer) an error recovery strategy based on selective repeat with Negative Acknowledgement (NACKs) scheme is used. In a NACK based scheme, the receiver explicitly indicates to the sender which segments went missing. The NACK based scheme eliminates the retransmission ambiguity problem, and also results in minimal control traffic overhead in the normal case.

Where possible, NACKs are piggybacked on to outgoing segments. If there is no outgoing traffic, an empty 'dummy' segment is created to carry the NACK. As a result of piggybacking, should a NACK be lost, the loss of the carrier segment will be noted, and the NACK re-transmitted with its carrier segment.

The link condition is verified periodically by setting the ACK-able header flag in an outgoing segment (creating a dummy segment if none already exists). The receiving host generates an explicit ACK response, in the same manner it would a NACK. ACK-able messages are generated every few seconds, thus the hosts can detect whether a serious link stall is being experienced. If the client receives no replies for 30 seconds, it attempts to disconnect ad re-attach to the GPRS network; experience shows that this action often brings the link straight back to life.

GPRSWeb uses a connection start-up method very similar to the TCP Accelerated Open (TAO) scheme developed for T/TCP [17], avoiding SYNs/ACK control packets. Each host remembers the segment number last received, and expects to receive the segment following. No handshake is needed, since numbering is assumed to continue from where it left off. Wherever it is necessary to start a new sequence (e.g. a host reboots), initial segments of the new sequence are tagged with the *deltas* between their sequence number, and the base of the new sequence. A host can therefore determine the new sequence base, and issue NACKs for missing segments, even if those missing started a new sequence.

Whereas TCP has to operate over links with widely varying qualities, GPRSWeb can make many more assumptions about the underlying network. Since the GPRS network already implements a mechanism for sharing bandwidth between users, there is no need for the GPRSWeb protocol to implement its own congestion avoidance mechanism. Instead, a simple credit-based flow control scheme suffices.

GPRSWeb initially gives each host credit equivalent to an estimated value of the bandwidth-delay product (BDP) of the link: no slow-start phase is employed. For 3+1 class GPRS devices this initial estimate is 10KB.

This level of outstanding credit is refined over time based on the measured RTT and throughput, typically from timing ACKable segments. The credit value used is set to be 10% higher than the measured RTT-throughput product, to allow the link to remain fully utilised in the presence of typical levels of jitter. Since the outstanding credit is capped in this manner, we avoid the excess queueing long-lived TCP flows cause, and ensure that the buffer residency in the GGSN remains low.

The protocol implementation provides a message queue based interface to higher layers: messages are placed in a queue for transmission and retrieved from a queue after receipt. Within the protocol stack, messages are serialised and split into segments before transmission, and segments reassembled into messages on receipt.

6.3 Caching

GPRSWeb implements an extended caching scheme intended to optimise the hit rate of the client cache, and thus minimise page download time and reduce bandwidth requirements. In terms of the freshness of pages actually returned for the user, the cache is no more aggressive than allowed by the normal HTTP algorithm, unless the GPRS link is currently down in which case the client proxy can be configured to return potentially stale data to allow something to be displayed.

The GPRSWeb extended caching protocol indexes objects by their SHA-1 fingerprint (content hash). A separate table is maintained that maps URLs to the respective Content Hash Key (CHK). This enables multiple URLs pointing to identical documents to be stored just once in the cache. In some dynamically generated web sites such identical mappings are commonplace, resulting in significantly improved hit rates.

The client cache is intended to replace the browser's persistent cache. During the course of proxy installation, the browser's persistent cache is disabled and flushed. The browser's in-memory cache is left enabled for performance reasons.

Each **Cache Entry** contains a document body and the time it was last used, stored in a file named by the document CHK. The **Cache Index** maintains an in-memory list of cache entry metadata. It is initialised with data read from the cache entries on disk, and is updated alongside the on-

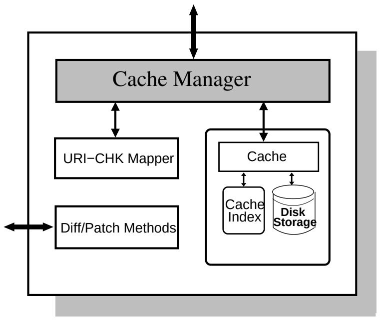


Figure 8: Cache Operation Overview

disk cache. The **URL Mapper** maintains mappings between URLs and the CHKs representing their bodies. Associated with each entry are the original HTTP Response headers, used to construct a Response when servicing a request from the cache. This allows multiple responses to share the same body, but with different headers.

When a URL mapping expires (as indicated by the conventional HTTP caching mechanism), it must be refreshed before being used again. The client proxy asks the server proxy to do this on its behalf.

The server proxy will check its own cache to see whether a more recent mapping exists. This can occur if another of its clients has accessed the same object. If not, it will have to contact the server to either check the modification time or fetch the object. Often, as a result of the pessimistic caching directives returned by sites with dynamic content, the object turns out to be identical to the previous version. The server proxy indicates this to the client by simply retransmitting the URL to CHK mapping, along with any caching directive returned by the server. In fact, thanks to the 'parse and push' mechanism described later the client often does not even need to request that a mapping be refreshed because the server proxy will have pro-actively sent a message containing the refreshed mappings.

The server proxy tracks the state of each client proxy's cache by modelling the replacement policy of the client, which is 'least recently used' eviction. The contents of its own cache is a superset of its clients'. If synchronisation is lost, for example, if the client proxy is directed to connect to a new server, the client uses low priority messages to update the server on its cache status.

6.4 Delta-encoding and Compression

The GPRSWeb proxies attempt to ensure that all data travelling over the GPRS link is in a compressed form, to reduce transfer size and improve response time. Unless the data is already in a compressed form (for example JPEG images or Zip archives), the gzip compression algorithm is employed.

Where the data being sent is an updated version of a previous object (same URL, different CHKs) a 'delta encoding' [15] algorithm is tried. Delta encoding sends differences between new and old versions of a document. The strategy is often very successful as many updated documents are very similar to their predecessor, particularly for dynamically generated content. A classic example is news site front pages that contain a string indicating the time of day.

Since the server proxy tracks the contents of the client cache is able to use the VCDiff [31, 32] algorithm to produce the *deltas* from a document it knows the client has. The deltas are gzip compressed and sent to the client if the resulting data is smaller than sending a compressed version of the new object.

Similar compression mechanisms are used for HTTP headers, both requests and responses. A separate string table is used for HTTP headers to avoid useful strings being evicted during the transfer of object data. This approach is very successful, since HTTP headers produced by modern browsers are rather verbose, and the variation between requests of the same type is small.

6.5 Parse-and-Push Operation

As discussed earlier, most web pages contain a number of images and other support objects (frames, style sheets etc.) that make up the page structure. These are eventually requested by the browser after parsing the HTML document.

The parse-and-push mechanism in the server proxy attempts to speculatively push toward the client objects and URL to CHK mappings that it knows are going to miss in the client cache. These are sent with lower priority than responses to requests explicitly made by the client. Responses are promoted if an explicit request is received for them. The main benefit from the parse-and-push mechanism is to keep the link utilized during delays due to the data dependencies between object references. For pages with complex layouts these can be quite significant – as well as the network RTT delay, it can take the browser some time to process the returned HTML.

The parsing performed by the server proxy is currently crude, but fast. Documents of type text/html are parsed using a regular expression to extract references to support objects. Duplicates are removed, and relative URLs combined with the document base to produce a list of candidate URLs. The server proxy may miss some object references (these will be simply be requested later by the client), and may in fact construct some invalid URLs. These will typically result in "URL not found" error codes when the proxy attempts to fetch them from the server, and will not be pushed to the client.

6.6 Image Transcoding

All the optimization techniques described up until now are "loss-less": they do not change the appearance of the page returned to the user. We have also experimented with a simple image transcoding scheme to shrink the size and quality of JPEG images sent over the wireless link.

Other groups have investigated the utility of transcoding, and mechanisms for its implementation far more thoroughly than us e.g. [11]. We included this functionality in the proxy as a placeholder for future work. In the experiments described here, the client proxy returns the image to its original width and height before passing it to the browser (though it is obviously somewhat degraded). A more complete implementation would degrade the image at the server proxy under the control of browser, using hints contained in the content.

7. GPRSWEB SYSTEM PERFORMANCE

7.1 Implementation

We have designed and implemented the GPRSWeb proxy system system over Windows XP/Windows 2000. The GPRSWeb server and client proxy was written in C# (Csharp) over Microsoft's .NET framework. C# is a new type-safe and garbage collected language with a powerful function library (especially for *networking*) to speed up project development. C# is also supported on WinCE based devices such as PDAs and smart-phones. We intend to port the client proxy code to such devices in the future. The client and server proxies share much of the source code for the GPRSWeb protocol stack and cache interface functionality.

7.2 Experimental Test Setup

Our experimental set-up for evaluating the GPRSWeb proxy system is shown in figure 9. The mobile terminal (laptop) was connected to the GPRS network via a Motorola T260 GPRS phone (supporting 3 downlink and 1 uplink channels). The GPRS network was provided by Vodafone. The GPRSWeb client software was installed on the mobile terminal, a laptop running Windows XP.

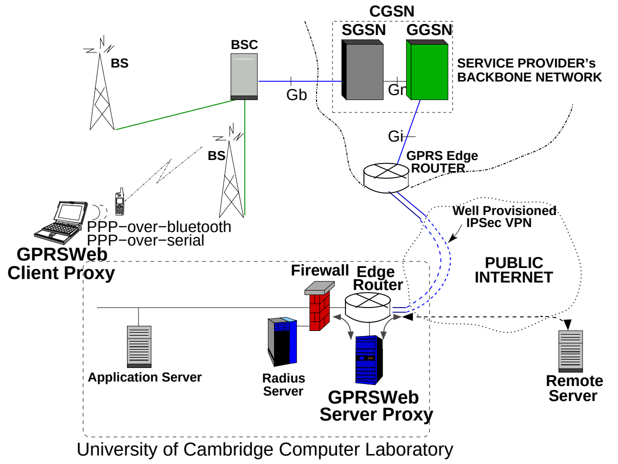


Figure 9: Experimental Test Bed Set-up

Since we were unable to install equipment to run the GPRSWeb proxy server next to the CGSN, we made use of a well provisioned IPSec VPN to 'back haul' GPRS traffic to the lab. The proxy server ran on a Windows 2000 server located near to the tunnel endpoint.

7.3 Experimental Results

In this section, we present an evaluation of how well the GPRSWeb proxy system improves WWW performance over GPRS. Specifically, we attempt to quantify the overall performance benefit relative to an unassisted browser. These experiments were performed using a single, stationary, mobile client to minimise variation in GPRS link performance.

We used the Mozilla 1.0 browser for recording these results. This was chosen due to the availability of source code, enabling us to instrument the browser to log web page download times. In non-persistent mode, Mozilla employs up to 8 parallel connections. In persistent connection mode, it uses up to 6 simultaneous connections. GPRS performance with other web browsers (Internet Explorer 6, Netscape 6) seems very similar to that of Mozilla, though we have not evaluated them as thoroughly.

7.3.1 *Test Web Sites*

To evaluate the performance benefits of our scheme, we performed experimental downloads of two synthetically arranged web pages offering static content, and also snapshot of the front page of two popular `news` web-sites and an `e-commerce` web site.

We arranged for these test pages to be hosted on a local server, eliminating the performance vagaries of public networks and servers. Furthermore we were able to control when content on these local pages was updated.

To create the synthetic pages, we adhered to the approach used in [19], composing a number of objects from other websites into a single page according to object type and file size distributions observed in HTTP proxy log traces. We consider two types of web site `STATIC-I` and `STATIC-II` (see table 1 and 2). The first is a relatively simple page consisting of a base HTML document with a few jpeg/gif images. The second one represents a more complex page comprising 46 objects.

For real-test web pages, we created mock-up of two popular news web sites, `www.cnn.com` and `www.bbc.co.uk`, and a third e-commerce web-site `www.amazon.com` based on a snapshot of their front page. We term them here as `1CNN`, `1BBC`, and `1AMAZON` respectively. These web-pages consisted of over 50 embedded objects including cascading style sheets (.css), active server pages (.asp) and java scripts (.jss).

Resource Type	Size/ Size Range	Total Number of files/objects	% w.r.t total content
index.html	17KB	1	52%
jpegs	2KB-4KB	3	29%
gifs	200B-5KB	5	19%

Table 1: `STATIC-I` web page composition

Resource Type	Size/ Size Range	Total Number of files/objects	% w.r.t total content
index.html	40K	1	25%
jpegs	2KB-5KB	8	17%
gifs	200B-2KB	24	17%
gifs	2KB-10KB	12	34%
gifs	>10KB	1	7%

Table 2: `STATIC-II` web page composition

7.4 Performance Evaluation

We present results comparing the performance downloading the test pages using the unassisted browser *vs.* using the GPRSWeb proxy. We used the default setting of the GPRSWeb proxy, that employs the full set of loss-less optimization techniques: *the enhanced transport protocol*, *data compression*, *delta encoding*, *extended caching* and *parse-and-push*.

Additionally, a separate experiment was performed with image transcoding also enabled. The transcoder module degraded only JPEG images, and only by a small amount, resulting in a typical image transfer size reduction of about 10%.

Except where stated, we flushed all client caches before each download test. We report results with both the server-side cache 'hot' and 'cold'.

We evaluated the following scenarios:

- `http-10:-` We measured download times using Mozilla over GPRS in non-persistent (HTTP/1.0) mode operating directly with the server.
- `http-11:-` These measurements were taken with Mozilla operating directly with the server in persistent connection (HTTP/1.1) mode.
- `gprsweb-1:-` These measurements are taken with the browser using the GPRSWeb proxy, but with cold client and server-side caches. Image transcoding was disabled.
- `gprsweb-2:-` Similar to `gprsweb-1`, but with a hot server-side proxy cache from which all objects are able to be served.
- `gprsweb-21:-` The scenario is similar to the `gprsweb-2`, but with image transcoding enabled.
- `gprsweb-3:-` Represents the best case scenario – a hit at the local client cache.

For each of the scenario discussed above, we recorded download times from 30 successful runs and plot the mean (average) value of the download times and corresponding standard deviation. For additional clarity, we also plot the median along with minimum and maximum value of the download time from each of the data set.

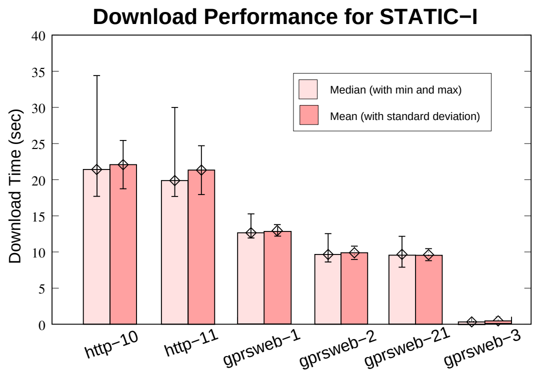
Figure 10(a) shows the mean download times for `STATIC-1`. We observe that use of HTTP/1.1 offers only meagre benefit (3-5%) over HTTP/1.0. However, we see a major improvement in download times using GPRSWeb protocol. Even with a cold server-side cache (`gprsweb-1`) the use of GPRSWeb protocol leads to 40%-45% reduction in mean page download times over GPRS. In the `gprsweb-2` case where the server-side cache is warm we record a performance improvement of 55%-60%.

Finally, the test where all objects hit in the local client proxy cache obviously gives the best performance. Page rendering time is broadly similar to that of a browser accessing content from its own local persistent cache.

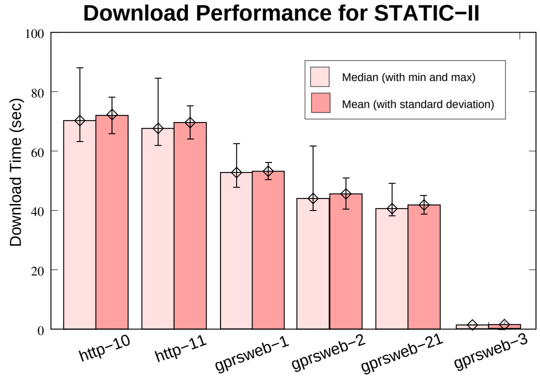
For the cold server-side cache case, the results from `STATIC-1` are reflected for other test pages – `1AMAZON` shows a mean reduction of about 45-50% while `1BCC` gives a benefit of over 50-55% (see figure 11 (a) and (b)). On the other hand, the benefits provided by GPRSWeb were not quite so substantial for `STATIC-II` and `1CNN`. In the cold server-side cache case GPRSWeb offers an improvement of 26% and 29% in page download times for `STATIC-II` and `1CNN` web pages respectively. With a warm cache, however, we see an encouraging reduction of 35-40% in mean page download time.

In these tests, our simple image transcoding optimization shows little benefit, and in fact the image processing delay actually makes matters worse in the `1CNN` case. In the `STATIC-II` page where there are several JPEG images some small advantage is shown. This is similar even for `1AMAZON` and `1BBC`. To properly evaluate the potential of image transcoding we should probably degrade GIF and PNG images as well as JPEGs, and use a more aggressive quality reduction settings.

Based on these results, we believe that the GPRSWeb proxy is capable of substantial reductions in web page download time. However, the experiments presented are yet to exercise the proxy's CHK-based caching or delta-encoding of

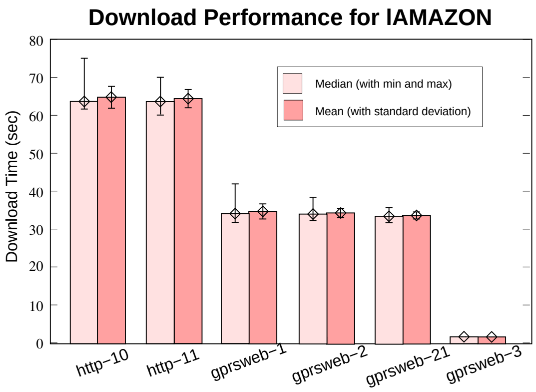


(a) STATIC-I

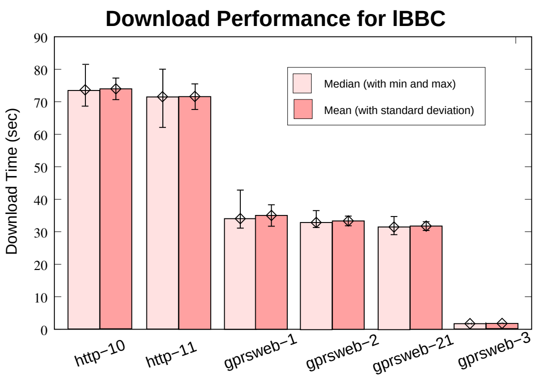


(b) STATIC-II

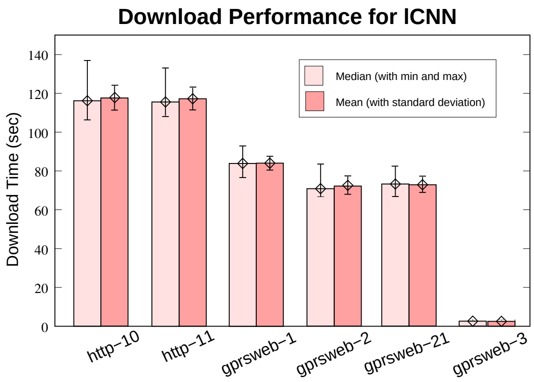
Figure 10: Mean download times with (top-down) (a) `STATIC-I`, (b) `STATIC-II` measured from 30 successful runs



(a) lAMAZON



(b) lBBC



(c) lCNN

Figure 11: Mean download times of some commercially available web-sites with (top-down) (a) `lAMAZON` (e-commerce), (b) `lBBC` (news), and (c) `lCNN` (news) measured from over 30 successful runs

object versions. This can only be done using a web site with changing, dynamically generated content. Experience using the proxy for real-life web browsing over GPRS suggest that these optimizations come into play quite frequently, and result in substantial performance wins when they do so.

We are in the process of building an experimental setup that enables recorded traces of user browsing behaviour to be accurately replayed, with the server reflecting the different versions of the content that were delivered on different occasions. This should enable an accurate evaluation of the real-world practical benefit of the extended caching and delta-encoding schemes.

8. RELATED WORK

A variety of previous research has examined techniques that can be used to optimise web browsing over high-latency links, but none has looked at GPRS links specifically. Related work can be broken into three categories: solutions that look at improving wireless web performance; transport enhancements that elevate TCP performance over wireless links, and finally other optimization schemes (e.g. smart caching, prefetching etc.) aimed at reducing data transferred and round trips made over the link.

8.1 Wireless Web Solutions

A number of significant research studies have investigated web performance, in general [18, 19, 15], and more specifically, wireless web performance [22, 12, 20, 21, 35].

Mogul *et al.* [18] discuss HTTP throughput and latency problems and show that each document and inlined image requires a minimum of ($2 \times$ RTT) for transfer. They propose the use of persistent connections and pipelining to improve overall web performance. Nielsen *et al.* [19] clearly demonstrate benefits of pipelined implementations, while Mogul *et al.* [15] argue that use of delta encoding and compression between client browser and a proxy can ‘remarkably’ improve web performance.

Liljeberg *et al.* [22] developed Mowgli Communications Architecture that uses a pair of proxies and employs its own protocol over the link. Mowgli is similar to GPRSWeb but focuses on GSM networks, and gives more weight to disconnected operation than we do for GPRS. It uses its proprietary protocol called Mowgli HTTP (MHTTP) that improves limitations of TCP and HTTP over GSM. The MHTTP protocol optimizes use of bandwidth using binary header encoding, data compression of text and images, and image transcoding.

WebExpress [12] from IBM focuses on improving web browsing performance over wireless links through caching, differencing, protocol and header reduction mechanisms. Unlike WebExpress, GPRSWeb focusses on the “always-on” GPRS link and specifically on web performance. This is somewhat different from WebExpress, which aims at improving web *form* performance on GSM links. WebExpress was designed to support applications characterized by repetitive and predictable responses, where only some certain information change in a given page.

Fleming *et al.* in [20] use a similar scheme to that used in

Wireless Web Solutions	Transport Protocol	Cach- -ing	Client prefetch	Parse- and-Push	Fast start and DNS Migration	Compre- -ssion	Filtering/ Transcoding	Delta-enconding and CHK
Mowgli [22]	Custom	√	×	×	√	√	√	×
WebExpress [12]	TCP	√	×	×	×	×	protocol/header reduction only	√ (no CHK)
MHSP [20]	TCP	√	√	×	×	×	√	×
FirstHop [35]	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
GPRSWeb	**Custom**	√	√	√	√	√	√	√

Table 3: Feature Summary of Wireless Web Solutions [`N.A. - Not Available`]

Mowgli, but also include prefetching schemes in their wireless world wide web proxy server and protocol using their new Multiple Hypertext Stream Protocol (MHSP). H. Balakrishnan *et al.* [21] use explicit loss notification (ELN) to improve web performance using a scheme by which senders are informed about the cause of loss event – network congestion or bit error. Thus ELN decouples sender retransmissions from congestion control.

Commercial products that improve upon the current GPRS performance are also available. A GPRS Accelerator under development by Firsthop [35] claims faster data transfers over GPRS. Their approach is to reduce data exchange and optimize protocols appropriately over the wireless link. However, it is unclear how this data reduction is achieved - using delta encoding and compression [15] or other sophisticated data (and header) compression techniques or by simply filtering/transcoding at the proxy. Nevertheless, their approach seems to necessitate a client-side software update. This obviously points us to believe that it might be making use of caching and/or delta compression/data transcoding.

8.2 Transport Layer Enhancements

In this section, we briefly review transport layer enhancements for wireless networks. Again, this can be factored into TCP, and non-TCP based solutions, which are generally UDP based. We briefly discuss some of these solutions to improve transport performance over wireless.

Snoop [23] is a TCP aware link-layer scheme that sniffs packets in the base station and buffers them. If duplicate acknowledgements are detected, incoming packets from the mobile host are retransmitted (if present) from the local cache. On the wired side, dupacks are suppressed from the sender, thus avoiding unnecessary fast retransmissions and the consequent invocation of congestion control mechanisms. However, Snoop was designed for wireless LANs rather than for 'long-thin' wide-area wireless links such as GPRS. As such, it does not address the problems of excess queueing at the base stations or proxies.

The 'split TCP' approach (such as the one used in [8]) is quite popular since it allows wireless losses to be completely shielded from the wired ones. I-TCP [24] uses a similar scheme for TCP over the wireless link, albeit with only some modifications. However, as TCP in I-TCP is not tuned for wireless links, it often leads to timeouts eventually causing stalls on the wired side and poor link utilization.

Then there are other schemes for improving TCP performance that make use of some sort of early warning signals. For example, Freeze-TCP [25] uses a proactive scheme in which a mobile host detects signal degradation and sends a Zero Window Probe (ZWP). The warning period – the time before which actual degradation occurs should be sufficient for the ZWP to reach the sender so that it can freeze its window. A drawback here is the reliability of this calculation and Freeze-TCP's inability to deal with sudden random losses. Furthermore, such schemes necessitate end-system changes.

Non-TCP based solutions typically optimize connection set-up/teardown and control overhead associated with TCP. One such example is Wireless Application Protocol (WAP) [28], in which a WAP gateway splits the transport with a new protocol for use over the wireless link. It uses two different protocols – Wireless Transport Protocol (WTP) and Wireless Datagram Protocol (WDP) over wireless, while using standard TCP over wired networks. WDP offers a general datagram service to the upper layer protocol to assist in communicating transparently with bearer services. Unlike WDP, WTP improves reliability of datagram services and relieves the upper layer from re-transmissions. It uses a message based transaction oriented protocol that assist in activities like web browsing.

8.3 Other Optimizations

A number of different studies have investigated cache performance. A hit at the client cache eliminates the need to traverse the high RTT wide-area wireless link. Some caching approaches are detailed in [26]. Cache Digests in [27] are a quick way of sending details about cache contents between caches – useful for synchronising caches over high latency links.

Web prefetching over networks, deterministic or predictive, is generally accepted to be useful. However, it is questionable if client-side predictive pre-fetching schemes over GPRS would be advantageous. The assumption made in earlier studies (e.g. Fleming *et. al.* [20]) was that since the link is idle anyway, why not use it for downloads, even if the prefetched page is oftentimes not useful. However, the links where this has been evaluated have employed *time-based* charging rather than the *volume* based charging typically used by GPRS operators. Since GPRS bandwidth is typically at least an order of magnitude more expensive than fixed-Internet bandwidth, the tradeoffs may be rather different.

The *parse-and-push* mechanism employed by GPRSWeb has some similarities to web prefetching, except the set of object pushed to the client are deterministic (and known to be of use to the client), and confined to support resources for the current page.

9. ISSUES AND DISCUSSION

In this section, we discuss some key issues relating to web optimization over GPRS.

9.1 Adaptation in Web Browsers

An issue crucial to WWW performance is how web browsers can be made to adapt to underlying network heterogeneity. As TCP connections continue to span a number of disparate links – wired as well as wireless – browser performance will increasingly depend upon how they attune their performance to adapt to the underlying network.

There are two approaches to optimizing browsing experience over heterogenous networks – browser adaptation, or a local proxy-based solution. With more penetration of WLANs and wide-area GPRS/3G, we feel that browsers of the future will have to be designed such that they can adapt to the underlying network. This can be a smart adaptation based on the type of the underlying network. Lower layers can indicate a change in the network, which in turn can assist browsers to quickly adapt to the new network.

9.2 GPRSWeb Server Proxy Location

In this section, we discuss possible locations for deployment of proxy servers within a GPRS network. A number of possibilities exist (see, figure 12): (a) close to the gateway router of the cellular provider where traffic density is likely high (in figure $Proxy(1)$), (b) near to the wired-wireless boundary (i.e. GPRS CGSN node or close) where traffic from a number of mobile hosts is aggregated (see $Proxy(2)$), (c) close to GPRS SGSN node where traffic from a number of base stations can be handled (shown as $Proxy(3)$), or (d) in the vicinity of the base station that handles traffic from a single cell (see, $Proxy(4)$).

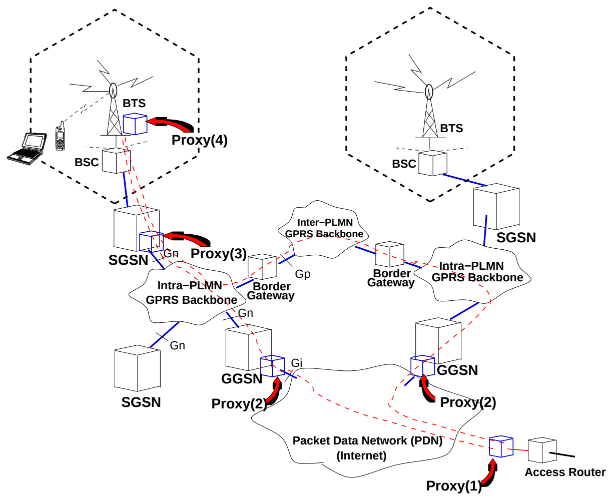


Figure 12: Possible Server Proxy Locations

In general, by placing proxies further inside a cellular network, one can get much better access to information about current radio link conditions than proxies located outside the network. Proxies located close to the base station can be informed of dynamically varying channel conditions, and could take action based on this. If fine-grained channel monitoring is not required, as in the case of GPRSWeb server proxy, it can be safely placed at, or close to the GPRS CGSN node, able to serve a large number of mobile clients from a single location.

9.3 Security Issues

GPRSWeb currently does not offer any additional encryption or authentication mechanisms. It relies on the security of the underlying GPRS network. A drawback using custom protocol over GPRS in GPRSWeb is that it can not provide end-to-end security (e.g. https). Designing a proxy-based solution that allows end-to-end security is still an open topic for research. However, protocols such as the Wireless Transport Layer Security (WTLS) protocol could be used to provide privacy, data integrity, and authentication for the link between the proxies. WTLS closely resembles Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol, yet is optimized for use over low bandwidth wireless links and for resource constrained mobile devices. WTLS supports datagram oriented protocols, so porting GPRSWeb would be straight forward.

Furthermore, the current GPRSWeb server proxy would be quite vulnerable to denial of service attacks, since it provides resource intensive services. The danger could be limited to some extent by ensuring that only clients from the wireless network can connect to the proxy, thus limiting the load an individual client can generate in an attempt to starve others.

10. CONCLUSION AND FUTURE WORK

In this paper, we explored the causes of TCP and HTTP under-performance in the GPRS environment. Due to GPRS' high latency, the need for latency mitigation was highlighted, leading to the design of the GPRSWeb proxy system, a pair of co-operating proxies positioned either side of the wireless link.

We have described our implementation and results of an initial performance evaluation of the prototype system. We have shown that the collective suite of optimization techniques implemented by GPRSWeb can lead to substantial reductions in mean page download times.

Currently, we have used the proxy system with only a limited set of concurrent clients. It would be interesting to perform tests (part of our ongoing work) to identify how well the server proxy scales to a wider client base. We are currently distributing GPRSWeb client software to a number of campus GPRS users. We are recording full tcpdump traces of all GPRS traffic generated by our user community, which will assist in evaluating system scalability and resulting user experience from using the proxy system.

In other ongoing work, we plan a thorough evaluation of the practical benefits of GPRSWeb's extended CHK-based caching and delta-encoding schemes. We intend to use our traces of user browsing activity and the corresponding server responses to accurately replay user activity, enabling us to precisely evaluate the performance gain these techniques can offer in the presence of real dynamically changing web content.

11. ACKNOWLEDGEMENTS

We would like to thank Vodafone Group R&D, Sun Microsystems Inc. and BenchMark Capital for supporting this work. Thanks also go to Tim Harris for his comments on this paper, and to Cristina-Ana-Maria Hristea for shepherding the submission of the final version of this paper. We also thank the anonymous reviewers for many constructive inputs. The complete source code for GPRSWeb including the test web-sites used in this paper is available:

`http://www.cl.cam.ac.uk/users/rc277/soft.html`

12. REFERENCES

[1] G. Brasche and B. Walke, "Concepts, Services and Protocols of the New GSM Phase 2+ General Packet Radio Service", *IEEE Communications Magazine*, August 1997.

[2] R. Chakravorty and I. Pratt, "Performance Issues with General Packet Radio Service (GPRS)", Journal of Communications and Networks (JCN), pages 266-281, Vol. 4, No. 2, December 2002 (ISSN 1229-2370). In the Special Issue of *Evolving from 3G deployment to 4G definition*
`http://www.cl.cam.ac.uk/users/rc277/gprs.html`

[3] R. Chakravorty, J. Cartwright and I. Pratt, "Practical Experience With TCP over GPRS", In *Proceedings of IEEE GLOBECOM 2002*, November 17-21, Taipei, Taiwan
`http://www.cl.cam.ac.uk/users/rc277/gprs.html`

[4] M. Meyer, "TCP Performance over GPRS", In Proceedings of IEEE WCNC, pages 1248-1252, 1999

[5] C. Bettssetter, H. Vogel, J. Eberspacher, "GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface", *IEEE Communication surveys* Third Quater 1999, Vol.2 No.3.

[6] R. Ludwig et al., "Multi-Layer Tracing of TCP over a Reliable Wireless Link", In Proceedings of ACM SIGMETRICS 1999.

[7] R. Chakravorty and I. Pratt, "WWW Performance over GPRS", In *Proceedings of IEEE International conference in Mobile and Wireless Communications Networks (IEEE MWCN 2002)*, September 9-11, Stockholm, Sweden
`http://www.cl.cam.ac.uk/users/rc277/gprs.html`

[8] R. Chakravorty, S. Katti, J. Crowcroft and I. Pratt, "Flow Aggregation for Enhanced TCP over Wide-Area Wireless", In Proceedings of INFOCOM 2003, San Francisco (to appear)
`http://www.cl.cam.ac.uk/users/rc277/gprs.html`

[9] H. Balakrishnan et al., "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM Trans. on Networking, Vol. 5, No.6, Dec. 1997.

[10] Z. Wang and P. Cao, "Persistent Connection Behaviour of Popular Browsers",
http://www.cs.wisc.edu/cao/papers/persistent-connection.html

[11] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, "Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspective", in IEEE Personal Communications, Vol. 5, No. 4, pages 10-19, August 1998

[12] B. C. Housel and D. B. Lindquist, "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment", In *Proceedings of ACM MOBICOM*, November 1996.

[13] M. Liljeberg et al., "Optimizing World-Wide Web for Weakly-Connected Mobile Workstations: An Indirect Approach", In *Proceedings of 2nd International Workshop on Services in Distributed and Networked Environments (SDNE)*, pages 132–129, Whistler, Canada, 1995

[14] J. C. Mogul, "Support for out-of-order responses in HTTP", *Internet Draft*, Network Working Group, 6 April 2001.

[15] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential benefits of delta encoding and data compression for HTTP", In *Proceedings of ACM SIGCOMM*, pages 181-194. Cannes, France, September 1997.

[16] R. Stewart et al., "T/TCP – TCP Extensions for Transactions", *Request for Comments (RFC) – 2960*, October 2000.

[17] R. Braden et al., "Stream Control Transmission Protocol", *Request for Comments (RFC) – 1644*, July 1994.

[18] V. N. Padmanabhan and J. C. Mogul, "Improving HTTP Latency", *Computer Networks and ISDN Systems*, vol. 28, pages 25-35, 1995

[19] H. Nielsen, J. Gettys, A Baird-Smith, E. Prud'hommeaux, H. Lie and C. Lilley, "Network Performance Effects of HTTP/1.1 CSS1, and PNG", In *Procedings of ACM SIGCOMM*, Cannes, France, September 1997

[20] T. B. Fleming, S. F. Midkiff, and N. J. Davis, "Improving the Performance of the World Wide Web over Wireless Networks", In *Proceedings of IEEE GLOBECOM*, 1997

[21] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance" In *Proceedings IEEE Globecom Internet Mini-Conference*, Sydney, Australia, November 1998.

[22] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen, "MOWGLI WWW Software: Improved Usability of WWW in Mobile WAN Environments", *IEEE Global Internet*, November 1996

[23] H. Balakrishnan, R. Katz and S. Seshan, "Improving TCP/IP performance over Wireless Networks", In *Proceedings of ACM MOBICOM*, November 1995

[24] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts", In *Proceedings of the 15th IEEE ICDCS*, pages 136-143, Vancouver, BC, May 1995.

[25] T. Go , J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments," In *Proceedings of IEEE INFOCOM 2000*, Israel.

[26] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques", *IEEE Communications Magazine*, Internet Technology Services, May 2000.

[27] L.Fan, P Cao, J. Almeida and A. Z. Broder, "Summary Cache: A scalable wide-area web cache sharing protocol", Technical Report 1361, Department of Computer Science, University of Wisconsin – Madison, Feb 1998.

[28] `http://www.wapforum.org`

[29] NIST, FIPS PUB 180-1: "Secure Hash Standard",
`http://www.itl.nist.gov/fipspubs/fip180-1.html`, April 1995

[30] NZipLib, The .NET Zip Library
`http://www.icsharpcode.net/OpenSource/NZipLib/`

[31] Internet Draft, "The VCDiff Generic Differencing and Compression Data Format",
http://www.ietf.org/internet-drafts/draft-korn-vcdiff-06.txt, November 2001

[32] JLibDiff, Java Diff Library
`http://sourceforge.net/projects/jlibdiff/`

[33] J. Cartwright, "GPRS Link Characterization",
`http://www.cl.cam.ac.uk/users/rc277/linkchar.html`

[34] "An Introduction to the Vodafone GPRS Environment and Supported Services", Issue 1.1/1200, December 2000, Vodafone Ltd., 2000.

[35] Firsthop GPRS Accelerator, `http://www.firsthop.com/`

[36] `tcpdump(http://www.tcpdump.org)`,
`tcptrace(http://www.tcptrace.org)`,
`ttcp+(http://www.cl.cam.ac.uk/Research/SRG/netos/netx/)`