Optimizing Web Delivery Over Wireless Links: Design, Implementation, and Experiences

Rajiv Chakravorty, Andrew Clark, and Ian Pratt

***Abstract*—World over wide-area wireless Global System for Mobile Communication (GSM) networks have been upgraded to support the general packet radio service (GPRS). GPRS brings "always-on" wireless data connectivity 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 show what causes the web and its underlying transport protocol TCP to underperform in a GPRS wide-area wireless environment. We examine why certain GPRS network characteristics interact badly with TCP to yield problems such as: link underutilization 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 a web optimizing proxy system called GPRSWeb that mitigates many of the GPRS link-related performance problems with a simple software update to a mobile device. The update is a link-aware *middleware* (a local "client proxy") that sits in the mobile device, and communicates with a "server proxy" located at the other end of the wireless 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 transfer of modified pages, DNS lookup migration, and a UDP-based reliable transport protocol that is specifically optimized for use over GPRS. We show that these enhancements results in significant improvement in web performance over GPRS links.

***Index Terms*—General packet radio service (GPRS), performance, proxy, web, wireless.**

I. Introduction

ALL OVER the world Global System for Mobile Communication (GSM) cellular networks have been upgraded to support the general packet radio service (GPRS). GPRS offers "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 [17], [26] on transmission control protocol (TCP) performance, but we have found actual deployed network performance to be somewhat different.

Some of the Web performance issues observed in GPRS are shared, to some extent, with wireless local area network (WLANs) like 802.11 $(WiFi)$, 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 some other 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:

- What are the "typical" GPRS network characteristics?
- What are the practical performance issues using TCP and HTTP over GPRS?
- What performance benefits can we achieve using various transport and application-level optimizations?

In short this paper presents our practical experiences over production GPRS networks, and our attempts to build a system that optimizes WWW performance over GPRS. After a brief overview, we summarize our work to characterize GPRS link behavior in Section II. Section III identifies particular problems experienced by TCP flows over GPRS, and Section IV examines how these are exacerbated by application-layer protocols such as HTTP.

In Section V, we present the design and implementation of our GPRSWeb proxy system—a Web optimizing dual-proxy system consisting of the client middleware (the client proxy) and a 'server proxy'. The client middleware is a local proxy that resides in the mobile device, while the GPRSWeb server proxy is located in the cellular network close to the wired-wireless border. The GPRSWeb system improves Web content delivery with an optimized transport protocol specifically tailored for GPRS wireless environments, an extended caching scheme, server controlled *parse-and-push* functionality, data (payload) compression, and delta encoding to improve performance in presence of dynamic Web content. GPRSWeb requires no additional instrumentation or modification to be made either to Web browsers, mobile clients, or content servers.

Section VI discusses GPRSWeb system performance and Section VII presents related work. We discuss issues related to GPRSWeb deployment in Section VIII and conclude in Section IX.

Manuscript received November 1, 2003; revised May 15, 2004.

The authors are with the Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, U.K. (e-mail: rc277@cl.cam.ac.uk; Rajiv.Chakravorty@cl.cam.ac.uk; Andrew.Clark@cl.cam.ac.uk; Ian.Pratt@cl.cam.ac.uk).

Digital Object Identifier 10.1109/JSAC.2004.839398

0733-8716/$20.00 © 2005 IEEE

II. GPRS Link Characterization

We have used a commercial GPRS testbed for link characterization. In this testbed, the mobile terminal (MT), e.g. a laptop, connects to the GPRS network through a GPRS-enabled interface—a PCMCIA GPRS card or a phone. In order to use the GPRS network, the MT first *attaches* itself to the gateway GPRS support node (GGSN) through a signaling procedure and establishes a point-to-point protocol (PPP) connection. The MT is dynamically assigned an IP address and the GPRS network is responsible for delivering data to and from this IP address as the MT moves through the network.

GPRS like other wide-area wireless networks, exhibits many of the following characteristics: low and fluctuating bandwidths, high and variable latency, and occasional link "blackouts" [31]. To gain clear insight into the characteristics of the GPRS link, we conducted a series of network link characterization experiments. These 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 found no major performance differences between the network operators, and variation between different handsets of similar GPRS device class is minimal. A more detailed description of how these tests were conducted (uplink/downlink latency measurements, tools used, etc.) can be found in [31]. We also provide a comprehensive description on GPRS link characterization in the form of a separate technical report in [15]. We enunciate some of our key findings.

High and Variable Latency: GPRS link latency is high and variable: 400–1300 ms in the uplink and 500–3000 ms in the downlink, as shown in Fig. 1. Round-trip times of 800 ms or more are typical. The variability seen in the link latency is due to the retransmissions (ARQ) at the radio link control (RLC) layer and depends on the wireless channel conditions. Fig. 1 shows significantly higher number of link-layer retransmissions occurred over the downlink (shows greater spread in the distribution). In contrast the uplink shows a much tighter delay distribution indicating better radio conditions. These measurements for delay distributions were obtained using a version of `ttcp` program (`ttcp+` [3]) modified to use precise timestamps between time-synchronized hosts in stationary conditions [31].

The link also shows a strong tendency to "bunch" packets—the first packet in a burst is likely to be delayed and experience much more jitter than the following packets. This indicates that a substantial proportion of the latency is incurred when the mobile terminal transitions from previously being idle [31]. Packets that are already queued for transmission can then follow the first out over the radio link without incurring additional jitter.

Fluctuating Bandwidth: We observe that signal quality leads to significant (often sudden) variations in perceived 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 (three downlink channels, one uplink), we observed a maximum downlink payload 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. Factors such as protocol overheads contribute to this discrepancy, see [26] and [31] for more information.

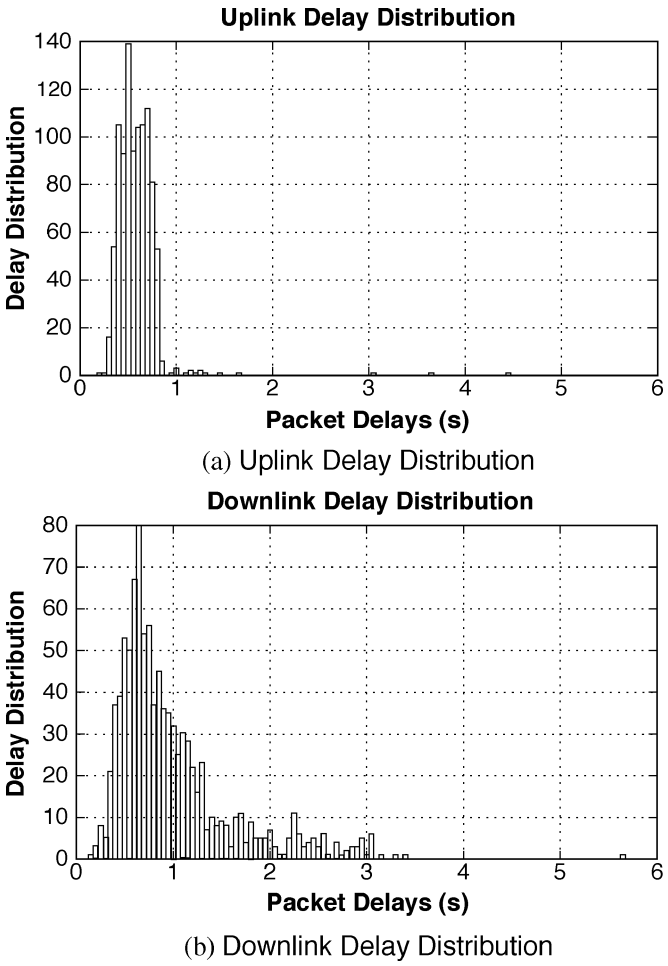

Fig. 1. Single packet time-in-flight sample delay distribution over GPRS links with plots showing (a) uplink delay and (b) downlink delay distribution for 1000 packet samples of size 1024 bytes each.

Packet Loss: The RLC layer in GPRS uses an automatic repeat request (ARQ) scheme that works aggressively to recover from link-layer losses. Thus, higher level protocols (such as IP) rarely experience noncongestive losses. Packets can be lost over the GPRS link during: 1) deep fading leading to bursty losses and 2) cell reselections resulting in a link "blackout" condition. In both cases, consecutive packets in a window are usually lost.

Link Outages: Link outages are more frequent when moving at speed or, for example, 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 40 s. Sudden signal quality degradation, prolonged fades and intrazone 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

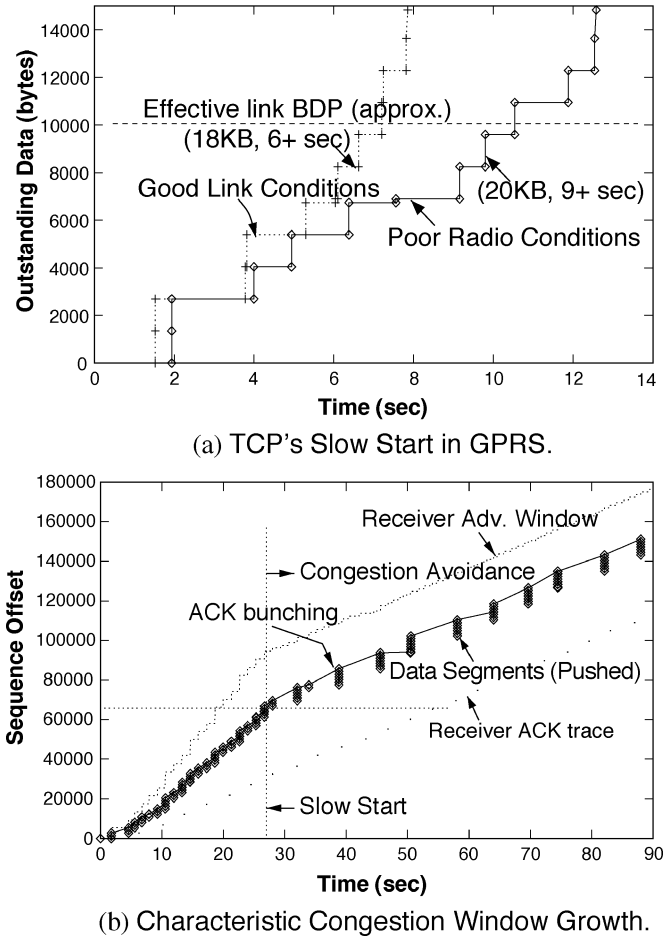

Fig. 2. (a) Shows that slow-start takes over 7 s to expand the congestion window sufficiently to enable the connection to utilize the full link bandwidth. (b) Captures the characteristic exponential congestion window growth due to slow-start. Maximum segment size (MSS) was set at 1400 bytes.

terminal would stall and stop listening to its temporary block flow (TBF) [11]. In such cases, we had to terminate and restart the PPP session.

III. TCP Performance Over GPRS

In this section, we concentrate on TCP performance issues over GPRS. We focus on connections where the majority of data is shipped in the downlink direction, as this corresponds to the prevalent behavior of most mobile applications, such as Web browsing, file download, reading e-mail, news, etc. A more comprehensive description on TCP performance problems over GPRS is available [31].

TCP Startup Performance: Fig. 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 10 KB. This estimate is approximately correct under both good and bad radio conditions, as even though the link bandwidth drops under poor conditions the RTT tends to rise. 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 s 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 18 KB, 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 very significant.

ACK Compression: A further point to note in Fig. 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 ACK's 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 Queueing: 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 Fig. 9). However, at the time these measurements were taken, we found that the existing GPRS infrastructure offered substantial buffering: UDP burst tests indicate that over 120 KB of buffering is available in the downlink direction. Most GPRS networks offer high buffering between 50–200 KB [31]. Hence, 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 64 KB 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. Fig. 3(b) shows a TCP connection in such a state, where there is 40 KB of outstanding data leading to a measured RTT of tens of seconds.

Excess queueing exacerbates other issues.

- `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 [36].
- `SYN timeout`. Excess queueing caused by long-lived flows results in attempts to establish new connections timing-out before completing the three-way handshake [36].
- `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 (sacks) or coarse timeouts in TCP over a saturated GPRS link takes many seconds. This is depicted in Fig. 3(a), where *drain time* is about 30 s.

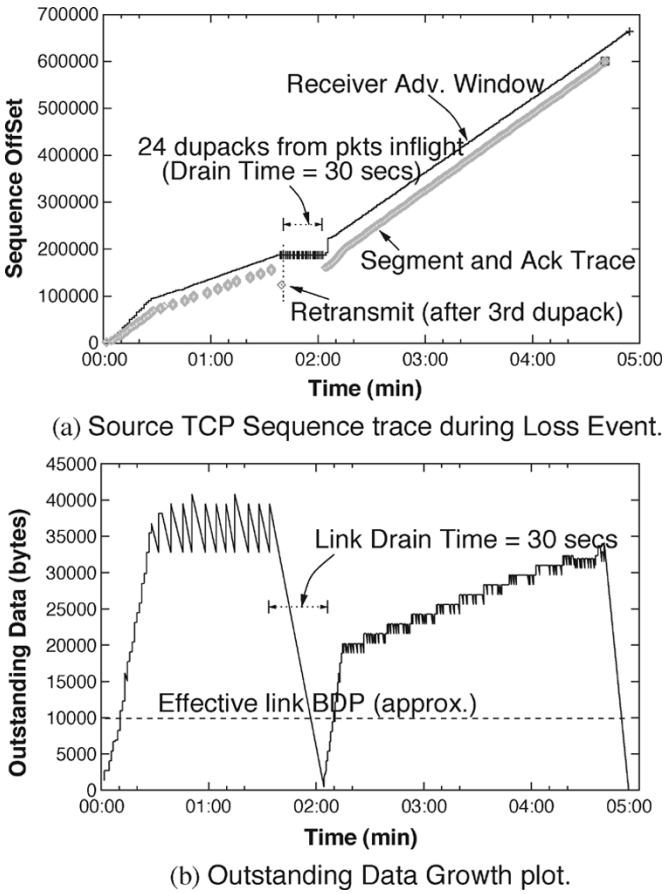

Fig. 3. Case of time-out due to a dupack(sack). (a) Shows the sender sequence trace. (b) Captures the corresponding outstanding data. MSS here was set at 1400 bytes.

TCP Loss Recovery Over GPRS: Fig. 3(a) and (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 emphasizes 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. Fig. 4 shows a file transfer f2 initiated 10 s 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 s) 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 [31]. The implicit favoring of long-lived flows often has the effect of delaying the "important" objects that the browser needs to be able to start displaying the

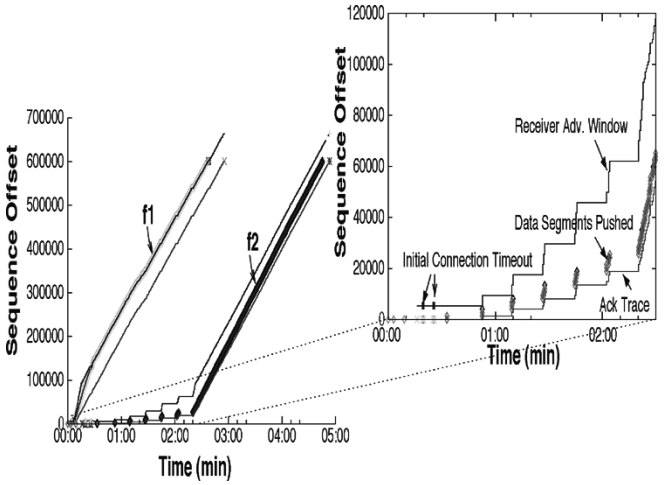

Fig. 4. Sequence plots of two concurrent file transfers over GPRS, where flow f2 (in close-up) was initiated 10 s after f1. MSS in this case was set at 1400 bytes.

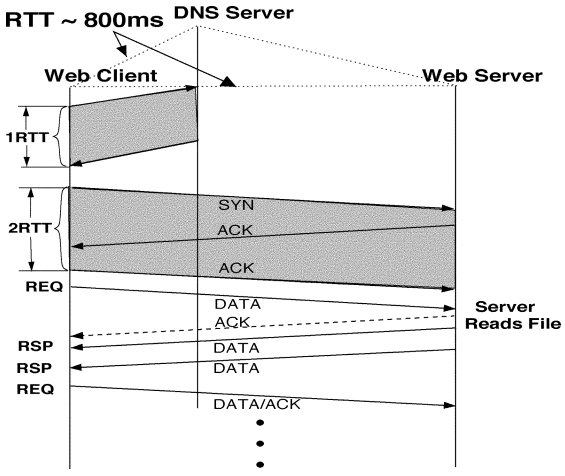

Fig. 5. Web connection overhead.

partially downloaded page, leading to decreased user perception of performance.

IV. WWW Performance Over GPRS

The results from the preceding section highlights many issues related to TCP performance in a wide-area wireless GPRS environment. In this section, we briefly review the key issues related to Web browsers, and specifically, Web performance over GPRS. More information related to Web performance issues over GPRS can be found in [31].

Typically, a Web connection could entail two round trips. As shown in Fig. 5—in the first round-trip, the 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. As RTTs over GPRS are high, frequent DNS lookups are costly as they must be completed before other work can continue. After resolving the DNS name, the Web client initiates a TCP connection with the remote server. As usual, every TCP connection will have to proceed through a three-way TCP handshake, which means that an extra RTT is incurred before the connection can be used. The impact of such RTTs is that it

induces "idle" times during downloads, resulting in link underutilization.

Popular Web sites usually contain embedded objects hosted under different domain names. In an attempt to improve end-user experience, these Web sites offload some of their static content to a set of servers located geographically closer to the users [13]. For example, news Web sites such as www.cnn.com contain embedded objects that point to many distinct domain names, e.g., Akamai content servers. Thus, when a browser performs DNS queries for such domain names, each query incurs a delay of at least one GPRS RTT. The situation is further exacerbated by content distribution networks that employ a smaller DNS time-to-live (TTL) value in their DNS responses for load balancing purposes [12], [27]. This results in browsers more frequently querying for DNS resolution.

Browser behavior is obviously crucial to better experience over any given network. Unfortunately, most current Web browsers are tuned for LAN environments and often perform poorly in a resource restricted setting. Web browsers (e.g., mozilla) open multiple concurrent TCP connections over a link simultaneously [31]. The inherent nature of TCP's congestion control algorithm implies that N connections will be N times more aggressive when compared with a single TCP connection 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) blocking problems [21]. An aggressive browser will obviously reap benefits over conventional high bandwidth links shared by many users.

However, using multiple TCP connections over "long-thin" GPRS links has its own drawbacks. First, protocol control (SYN/ACK/FIN's) overhead associated with greater numbers of connections is high. This is further exacerbated by the overhead of the protocol headers (55 bytes as in [6]) for data packets that are exchanged over the link. Second, the three-way handshake delay while establishing a TCP connection can be significant due to the high latency of GPRS links. Furthermore, 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 queueing 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 time-outs. Experiments in [31] over production GPRS networks show that aggressive Web browsers at times saturate the downlink GPRS GGSN buffers.

There exists a tradeoff in the number of simultaneous, persistent, TCP connections to use over GPRS [32]. If Web browsers open few persistent TCP connections then it may lead to link underutilization. Opening many increases connection setup overhead and consequently degrades performance. Experience shows that support for persistent connections is not always implemented in Web server software or can sometimes be deliberately turned off [13]. In fact, many commercial Web servers explicitly close (FIN) connections after certain number of requests. This implicitly forces Web clients to open many new TCP connections to download the entire content. Unfortunately this behavior degrades end-user Web experience over GPRS.

The full benefits of HTTP 1.1 cannot be realized without making use of HTTP *pipelining*, where multiple outstanding requests are permitted on the same connection [29], [34]. Without pipelining, a RTT delay is normally incurred between objects on the same connection, and worse, slow-start must be performed at the start of every object (i.e., HTTP GETs) since the connection will have gone idle. Unfortunately, support for pipe lining is currently rare. Our experiments also show that HTTP request pipe lining improves Web performance over GPRS [31], [32].

V. The GPRSWeb Proxy Model

The previous section identified the causes of poor Web performance in a GPRS environment, and we now report on our attempts to overcome them. Clearly, performance can 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 widespread deployment.

In earlier works, we improved Web delivery over GPRS by focussing on transport TCP performance [33]. By installing a *transparent* TCP enhancing proxy in a cellular network, we improve its performance over wireless GPRS links. While this approach certainly benefits TCP flow performance in the downlink, there are some issues it is not able to address.

- **TCP connection setup overhead**: Many Web browsers continue to use HTTP 1.0 by default and open multiple TCP connections to download Web content. Using a TCP optimizing proxy is not able to eliminate the TCP three-way connection setup overhead in HTTP 1.0 when downloading the Web content.
- **Distributed structure of Web content**: Popular Web content is often located in different content servers each having a distinct public domain name. Downloading these Web pages incurs additional DNS lookups for resolving these domain names of different content servers [13]. Furthermore, Web browsers may also open multiple TCP connections to each content server thereby sometimes increasing the overall connection setup overhead.

To overcome such limitations of Web content distribution, our approach in GPRSWeb 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 implements a number of techniques to enhance performance. This enables GPRSWeb to be both browser, as well as 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 to mitigate the effects of high RTT in GPRS. A custom middleware ("client-side" proxy) must be installed on the mobile client device. As part of the installation, the Web browser is configured to route all HTTP requests through this middlware-based proxy

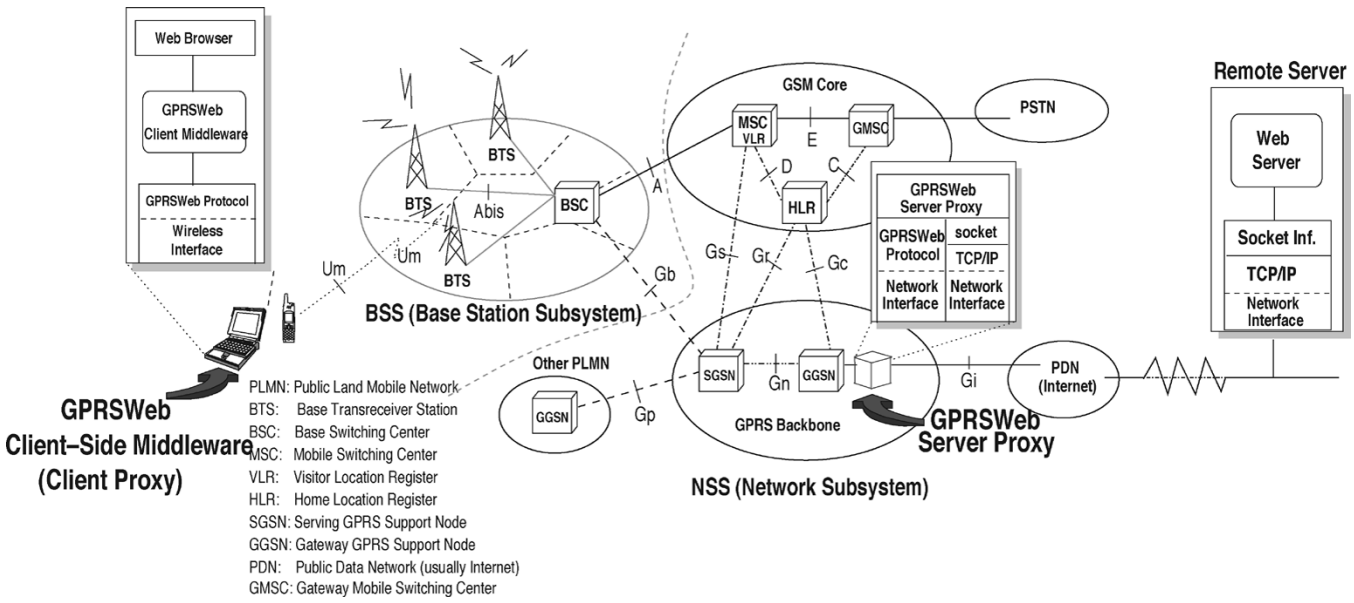

Fig. 6. GPRSWeb system architecture and components.

via a local TCP connection using the traditional *loopback* interface. As shown in Fig. 6, the client proxy communicates with a "server proxy," located on the other end of the GPRS link. The client proxy avoids costly DNS resolutions over the GPRS link using DNS lookup migration. In this approach, the client directly sends the compressed URL to the server proxy to resolve. The server proxy makes requests to the wired network on behalf of the client, and sends back responses. The cache content in the server proxy is shared, capable of servicing large numbers of mobile clients simultaneously.

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 located on either side of the GPRS link. Instead, we use a custom transport protocol (which we call the GPRSWeb protocol hereafter) that runs over UDP and implements ordered, reliable, message transfer. The protocol is optimized for GPRS link characteristics, and minimizes link traversals and responds efficiently in the event of the patterns of packet loss 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 [5] 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 shared 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 compresses 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 [7] algorithm to encode the differences between the old and new objects. The compressed *deltas* [22] 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 proactively starts *pushing* objects toward the GPRSWeb client cache if the link would otherwise be idle.

Our design of GPRSWeb is somewhat similar to the Mowgli Communication Architecture that also makes use of a pair of proxies and employs its own custom protocol over GSM wireless link [25]. Mowgli focuses on GSM networks and is primarily designed to handle lengthy link stalls. Other works in this direction is the IBM's WebExpress [14] that was designed to improve Web browsing performance over such wireless links. WebExpress improves performance through caching,

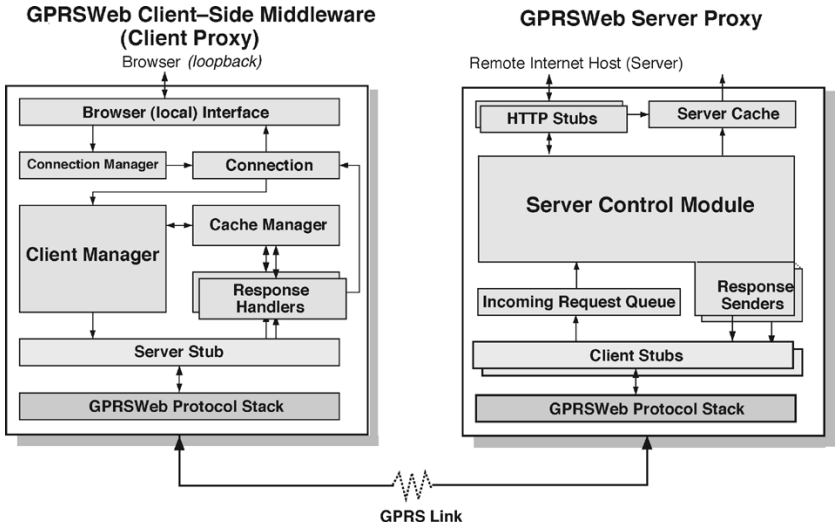

Fig. 7. Client-side middleware and server proxy structure.

differencing, protocol, and header reduction mechanisms. It was designed specifically for Web forms—applications characterized by repetitive and predictable responses where only some certain information would change in a given Web page. Similarly, Fleming *et al.* in [38] implement prefetching schemes in their wireless world wide Web proxy server and a new multiple hypertext stream protocol (MHSP).

However, none of these above solutions support schemes implemented in GPRSWeb such as CHK-based caching, server based parse-and-push, delta-encoding, fast start, and DNS lookups migration. As we shall demonstrate in Section VI, use of such schemes leads to significant performance benefits in Web browsing experience over high-latency wireless links.

A. 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. Fig. 7 show the components used in the client and the server.

The **connection manager** accepts TCP connections from the Web browser and passes them to a **connection** object. This queries the **client cache manager** (Fig. 8), 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 the message type. Object request messages are processed by the **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 coordinate data compression and other optimizations before the response is finally sent back to the client proxy.

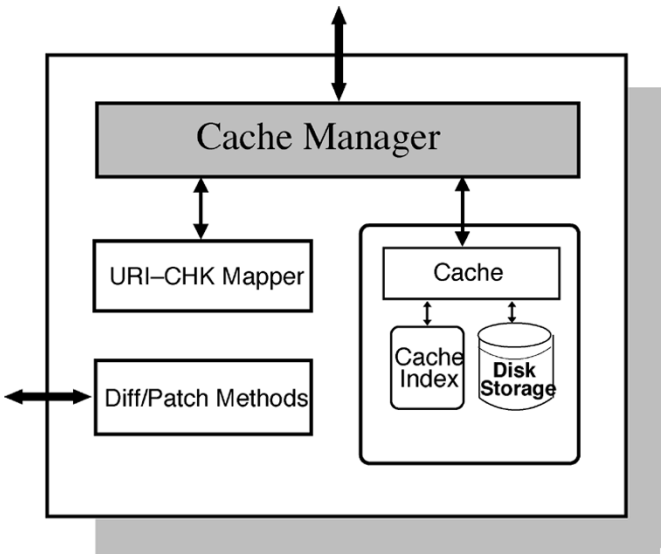

Fig. 8. Cache operation overview.

B. 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 optimize performance.

The basic unit of transfer is the *segment*, each of which is carried in a separate 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 relatively reliable RLC layer) an error recovery strategy based on selective repeat with negative acknowledgments (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 the 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 retransmitted 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 and reattach to the GPRS network; experience shows that this action often brings the link straight back to life. Note that this same technique helps to recover from link outage conditions caused by cell-reselections that may last up to few seconds.

GPRSWeb uses a connection start-up method very similar to the TCP accelerated open (TAO) scheme developed for T/TCP [30], 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 10 KB. This level of outstanding credit is refined over time based on the measured RTT and throughput, typically from timing ACK-able segments. The credit value used is set to be 10% higher than the measured RTT throughput product, to allow the link to remain fully utilized 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 serialized and split into segments before transmission, and segments reassembled into messages on receipt.

C. Caching

GPRSWeb implements an extended caching scheme intended to optimize the hit rate of the client cache and, thus, minimize 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 URL's pointing to identical documents to be stored just once in the cache. In many dynamically generated Web sites such identical mappings (known as response *aliasing* [40]) are commonplace, resulting in significantly improved hit rates. CHK-based caching offers *alias* protection. Thus, CHK-based caching offer gains not only in terms of storage at the server proxy, but also in the amount of data being transferred over GPRS.

The client cache is intended to replace the browser's persistent disk 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 initialized with data read from the cache entries on disk, and is updated alongside the on-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 proactively sent a message containing the refreshed mappings.

The server proxy tracks the state of each client proxy's cache by modeling the replacement policy of the client, which is "east recently used" eviction. The contents of its own cache is a superset of its clients'. If synchronization is lost, for example, if the client proxy is directed to connect to a new server, the client

could potentially use low priority messages to update the server on its cache status, but this is yet to be implemented.

D. 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" [22] 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 current time of day.

Since the server proxy tracks the contents of the client cache is able to use the VCDiff [7] 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.

E. 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.

F. 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., [10]). 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 the browser, using hints contained in the content.

VI. GPRSWeb System Performance

A. Implementation

We designed and implemented the GPRSWeb proxy system over Windows XP/2000. The GPRSWeb server and client middleware was written in C# (Csharp) over Microsoft's.NET framework. The complete source code for the implementation is publicly available [1]. 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.

B. Experimental Test Setup

Our experimental test setup for evaluating the GPRSWeb proxy system is shown in Fig. 9. The mobile terminal (laptop) running GPRSWeb client was connected to the Vodafone UK's commercial GPRS network via a Motorola T260 GPRS phone (supporting three downlink and one uplink GSM slot). Additionally, since we were unable to install the GPRSWeb proxy server equipment to run next to the Vodafone CGSN, we made use of a well provisioned IPSec VPN to "back haul" GPRS traffic to our lab. The proxy server ran on a Windows 2000 server located near to the tunnel endpoint.

Most current GPRS networks (including Vodafone UK's GPRS network that we use) make use of the static $\mathsf{CS}-2$ coding scheme for Forward Error Correction (FEC) [33]. CS-2 is a "good compromise" coding scheme [35]. With this scheme a "$3+1$" GPRS phone can support a maximum ideal downlink data rate of 39.6 Kb/s. Note that the actual payload throughput seen by the RLC layer will be somewhat less than the "ideal" downlink data-rate. In these experiments, the RLC link-layer reliability (ARQ retransmissions) is kept enabled.

C. Experimental Results

We present an evaluation of how well the GPRSWeb proxy system improves WWW performance over GPRS. Specifically, we attempt to quantify the overall performance benefits relative

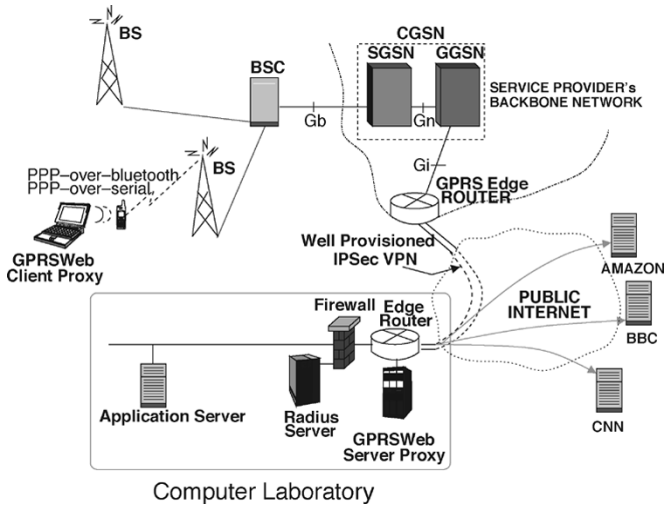

Fig. 9. Experimental testbed setup.

to an unassisted browser. These experiments were performed using a single, stationary, mobile client to minimize variation in GPRS link performance.

We used the Mozilla 1.0 browser in these experiments. Mozilla was used because its source code is freely available, enabling us to instrument the browser to log download times of the Web page. In HTTP 1.0 nonpersistent connections mode, Mozilla employs up to eight parallel simultaneous connections to each Web server. In HTTP 1.1 persistent connection mode, it can open up to four simultaneous TCP connections when configured to use a proxy. In our experience, GPRS performance with other Web browsers (Internet Explorer 6, Netscape 6) was also roughly similar to that of Mozilla. During these experiments, we also measured the signal quality at the location of the mobile client. Our measurements indicated receiver signal strength indication (RSSI) values between -95 dBm and -68 dBm and bit-error rate (BER) between 0%–4%, respectively. These values indicate moderate to good radio channel conditions.

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 an approach used in [20], composing a number of objects from other Web sites 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 Tables I and II). 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 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 `LCNN`, `LBBC`, and `LAMAZON`, respectively.

TABLE I
STATIC-I WEB PAGE COMPOSITION

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 II
STATIC-II 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%

These Web pages consisted of over 50 embedded objects including cascading style sheets (.css), active server pages (.asp), and java scripts (.jss).

D. 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 nonpersistent (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 value of the download times and corresponding standard deviation. For

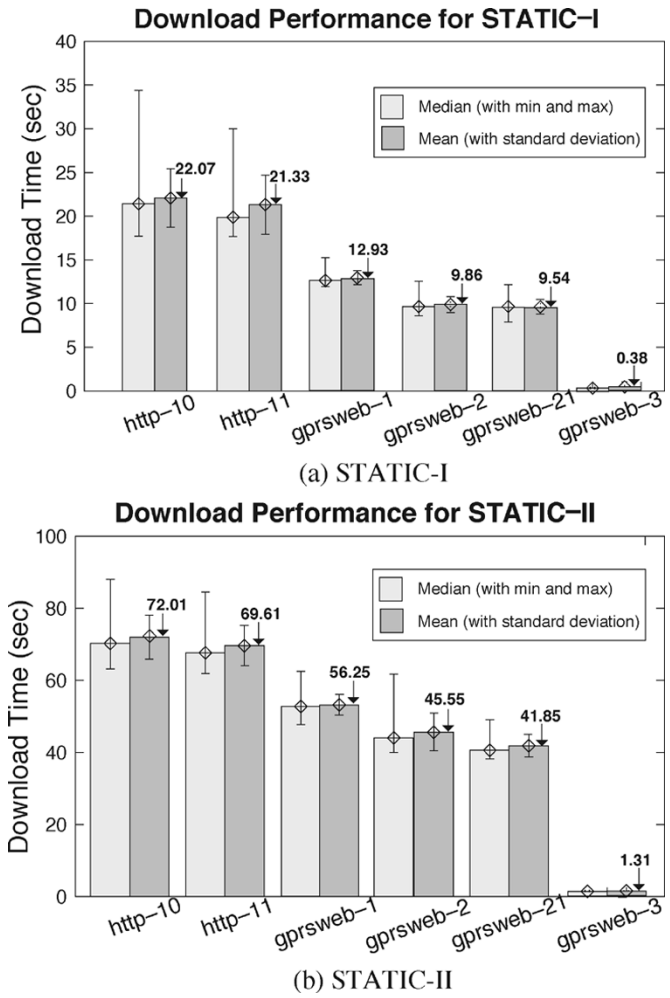

Fig. 10. Mean full Web page download times with (a) STATIC-I and (b) STATIC-II measured from 30 successful runs.

additional clarity, we also plot the median along with minimum and maximum value of the download time from each data set.

Fig. 10(a) shows the mean download times for STATIC-1. Here, 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—LAMAZON shows a mean reduction of about 45%–50%, while LBCC gives a benefit of over 50%–55% [see Fig. 11(a) and (b)]. On the other hand, the benefits provided by GPRSWeb were not quite so substantial for STATIC-II and LCNN. In the cold server-side cache case, GPRSWeb offers an improvement of 26% and 29% in page download times for STATIC-II and

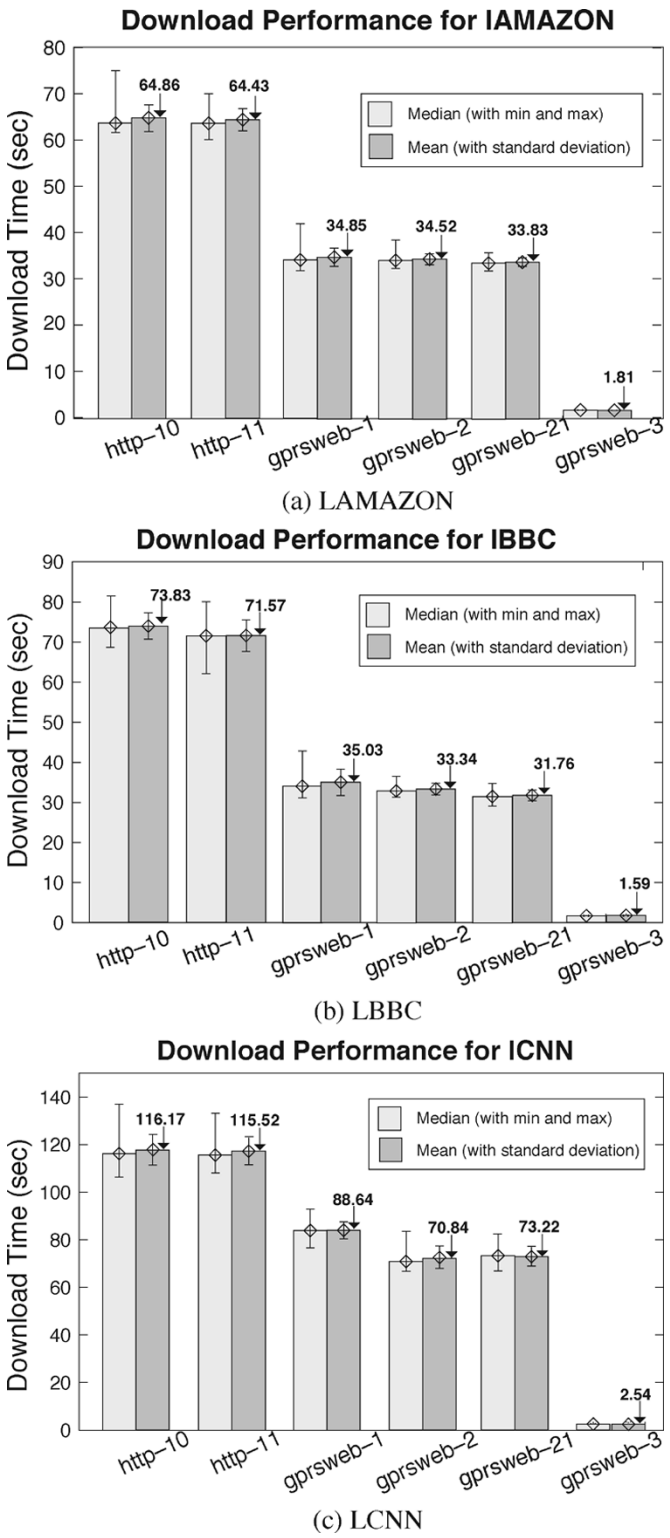

Fig. 11. Mean Web page download times of some commercially available Web- sites with (a) LAMAZON (e-commerce), (b) LBBC (news), and (c) LCNN (news) measured from over 30 successful runs.

LCNN 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, it seems that the image processing delay actually makes matters worse in the LCNN case.

TABLE III
WIRELESS WEB SOLUTIONS

Wireless Web Solutions	Transport Protocol	CHK Caching	Client Prefetch	Parse-and-Push	Fast start + DNS Migration	Compression	Filtering + Transcoding	Delta-encoding
Mowgli [25]	Custom	×	×	×	×	√	√	×
WebExpress [14]	TCP	×	×	×	×	×	×	√
MHSP [38]	TCP	×	√	×	×	×	√	×
FirstHop [2]	?	×	√	×	×	√	√	×
NetGain [8]	?	×	√	×	×	√	√	×
GPRSWeb	**Custom**	√	×	√	√	√	√	√

In the STATIC-II page where there are several JPEG images some small advantage is shown. This is similar even for LAMAZON and LBBC. An *aggressive* quality reduction settings for images may provide additional benefits [27], [41].

Our next set of experiments evaluate benefits from parse-and-push. To evaluate the extent of benefits available from using *parse-and-push*, we downloaded our sample test Web pages by disabling this feature in GPRSWeb. Our observations indicates that use of parse-and-push leads to an additional 4%–10% benefit in downloads of our example Web sites.

We now examine results for GPRSWeb's CHK-based caching and delta-encoding of the different object versions. This is possible using Web sites offering dynamically generated Web content. Hence, in this case, we directly make use of the actual (real) Web sites offering dynamic Web content. In these experiments, we repeatedly download the front pages of our Web sites every hour for three consecutive days. To explicitly quantify the benefits resulting from CHK-based caching and delta-encoding, we disable use of GPRSWeb proxy's data compression and parse-and-push feature in these tests. We then compute mean Web page download times from over ten successive download runs.

For the case of delta-encoding, we compare the mean Web page download times from the previous download runs (download conducted every hour), while for CHK-based caching, we compared mean download times for tests conducted simultaneously, with and without using this feature. *For typically fast changing Web sites such as CNN and BBC, we find on average an additional between 3%–8% improvement in Web download times from CHK-based caching and delta encoding*. The time downloading these Web pages are fast since images are already available in the server proxy cache, and html document size is small compared with the inlined images that remain typically unchanged. Between CHK-based caching and delta encoding, the benefit is mainly due to delta-encoding, except for the case of BBC where the benefit is somewhat higher using CHK-based caching. Results from live download tests also confirmed evidence of greater response aliasing in case of BBC at the time of these tests.

In general these results will vary for different Web sites. A real-world analysis of the practical benefits of delta-encoding and CHK-based caching would need a more thorough (empirical) evaluation. We are in the process of building a setup that enables recorded traces of user browsing activity to be accurately replayed, with the server reflecting the different versions of the content to be delivered on different occasions. This should enable an accurate evaluation of the real-world practical benefits of the extended caching and delta-encoding schemes. Using the proxy for real-life Web browsing over GPRS, confirms that these optimizations come into play quite frequently, and can result in significant bandwidth savings when they do so.

Our tests were conducted in a stationary environment to minimize link variations and avoid vagaries of the harsh mobile exteriors. However, our evaluations are also similarly applicable even for mobile environments.

VII. RELATED WORK

Commercial products that improve Web browsing performance over wireless links are available. A GPRS Accelerator by Firsthop [2] claims faster data transfers over GPRS. Their approach is to reduce the amount of data exchanged and optimize protocols over the wireless link. Other products include NETGAIN from FlashNetworks [8]. The *client-based* solution from NETGAIN is similar to GPRSWeb, but uses a modified HTTP protocol (with HTML reformatting), aggressive image compression and prefetching for improved Web performance. A feature-based comparison is given in Table III.

Past research has extensively explored improving transport performance over wireless. Examples of this include Snoop [19], I-TCP [9], etc. However, these schemes are meant for wireless LANs rather than cellular wireless links. Balakrishnan *et al.* [18] make use of explicit loss notification (ELN) to improve Web performance. Using ELN senders can be informed about the cause of loss event—network congestion or radio error. Thus, ELN decouples sender retransmissions from TCP's congestion control. For cellular environments WTCP [28] and *Ack Regular* [24] schemes have been proposed that benefits TCP performance over CDPD and CDMA 2000 3G-1X links, respectively. Some other schemes for improving TCP performance, for example, Freeze-TCP [39] uses a proactive scheme in which a mobile host can detect signal degradation and send zero window probes (ZWP). However, the warning period—the time before which actual degradation occurs should be sufficient for the ZWP to be able to reach the sender so that it can freeze its window.

Custom transport protocols optimize connection set-up/teardown and control overhead associated with TCP connections. One such example is the wireless application protocol (WAP) [4], in which a WAP gateway splits the transport with a new protocol for use over the wireless link. Note that the latest WAP 2.0 specification also includes "wireless profiled" TCP and HTTP protocols. While use of a gateway is optional in WAP 2.0, our experience with GPRSWeb demonstrates the potential benefits using a performance proxy in cellular environments. Hence, many optimizations discussed in this paper, e.g., CHK-based

caching, delta-encoding, parse-and-push, fast-start, etc., are applicable even for WAP 2.0.

Proxy and caching techniques have been extensively studied in the context of wired (e.g., dial-up) environments. Some caching approaches are detailed in [16]. Cache digests in [23] are a quick way of sending details about cache contents between caches—useful for synchronizing 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 prefetching schemes over GPRS would be advantageous. The assumption made in earlier studies (e.g. Fleming *et al.* [38]) 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 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 Web page.

VIII. Issues and Discussion

In this section, we discuss issues relevant to GPRSWeb proxy deployment in cellular data networks.

Server Proxy Location: The location of server proxy in a cellular data network can impact a number of service-specific issues—user mobility, transport protocol agility, server scalability, as well as service reliability. There are different locations where a GPRSWeb proxy server could be deployed (shown in Fig. 12): 1) close to the gateway router of the cellular provider where traffic density is likely high (in figure `Proxy1`); 2) near to the wired-wireless boundary (i.e., GPRS CGSN node or close) where traffic from a number of mobile hosts is aggregated (labeled as `Proxy2`); 3) close to GPRS SGSN node where traffic from a number of base stations can be handled (shown as `Proxy3`); or 4) in the vicinity of the base station that handles traffic from a single cell (marked as `Proxy4`).

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 the dynamically varying channel conditions, and could take action accordingly. If fine-grained channel monitoring is not required, 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. *By taking advantage of the server proxy location, a cellular operator can manage uninterrupted proxy service availability for mobile users in their networks.*

Proxy Server Scalability: GPRSWeb proxy server is expected to serve traffic from several mobile clients. However, if load on the proxy server increases drastically, it can impact the overall system performance. Therefore, appropriate load balancing schemes should be employed by cellular providers to address scalability issues related to proxy servers.

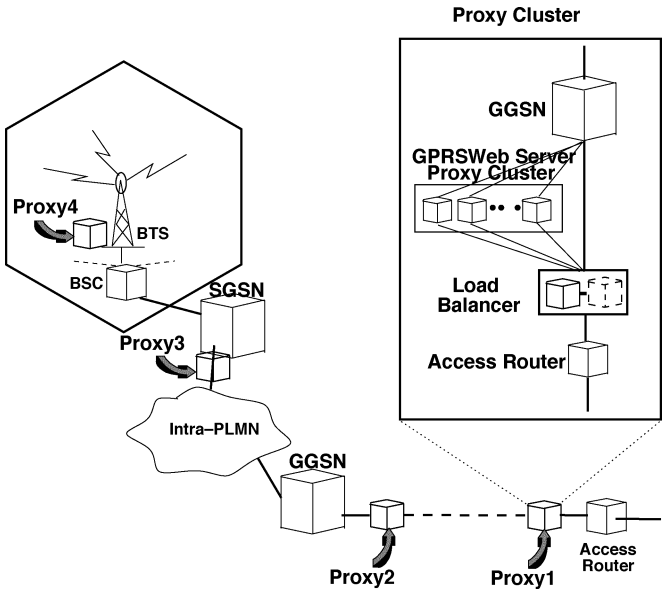

Fig. 12. Possible server proxy locations.

A client and server proxy in GPRSWeb maintains "persistent" association during a Web browsing session. This requires that any load balancing scheme employed by the cellular operators should appropriately configure traffic from a mobile client to be guided to the same proxy server within the proxy cluster—as long as the server remains available and the client active. A mobile client is switched to a different proxy server only when the original server is no longer available (e.g., shut down, crashed, etc.) or if there is no traffic anymore. Furthermore, a *backup* load balance switch may be employed for greater reliability. In this way an operator can overcome scalability issues related to GPRSWeb server proxy, uninterrupted service availability and reliability. Each proxy in the cluster pool will now handle traffic from a select set of mobile clients to limit the load from the mobile users.

Security Issues: The way proxy servers are deployed may have security-related implications. Web proxy-system such as GPRSWeb can be deployed either *explicitly* or *transparently*. Explicit deployment introduces some vulnerability by exposing proxy server IP address to Web clients (or the "client-side" proxy) to allow it to interact with the server proxy.

The use of custom protocol in GPRSWeb also requires additional mechanisms for security. Protocols such as WAP-based wireless transport layer security (WTLS) protocol can be used to provide privacy, data integrity, and authentication over the wireless link [4]. WTLS closely resembles secure socket layer (SSL)/transport layer security (TLS) protocol, yet is optimized for use over low bandwidth wireless links and suits resource constrained mobile devices.

A proxy server like GPRSWeb is quite vulnerable to denial of service attacks, since it provides resource intensive services. This potential security hazard can 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. Other architectural and

related policy issues also needs careful consideration when deploying proxies as it impacts the overall reliability of a solution. Internet RFC 3238 [37] deals with many such architectural and policy issues for proxy deployment in the Internet.

IX. Conclusion

Our work has explored the causes of TCP and HTTP under performance in a GPRS environment. Due to high latency in GPRS links, the need for latency mitigation was highlighted, leading to the design of the GPRSWeb proxy system, a pair of cooperating proxies positioned either side of the wireless link.

We have described our implementation and results of a 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.

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 the future, we also 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 gains these techniques can offer in the presence of real dynamically changing Web content.

Acknowledgment

The authors would like to thank Vodafone Group R&D, Sun Microsystems, Inc., and BenchMark Capital for supporting this work. Thanks also to T. Harris and J. Crowcroft for providing insightful comments on this paper. The complete GPRSWeb source code and test Web sites used in this paper are available from the COMS Project Web site: http://www.cl.cam.ac.uk/coms/.

References

[1] Cambridge open mobile systems (COMS) Project. [Online]. Available: http://www.cl.cam.ac.uk/coms/

[2] First-hop GPRS accelerator. [Online]. Available: http://www.firsthop.com/

[3] tcpdump. [Online]. Available: http://www.tcpdump.org; (tcptrace) [Online]. Available: http://www.tcptrace.org; (ttcp+) [Online]. Available: http://www.cl.cam.ac.uk/netos/netx/

[4] The WAP Forum Web-Page. [Online]. Available: http://www.wapforum.org

[5] NIST FIPS PUBS 180-1: Secure Hash Standard (1995, Apr.). [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.html

[6] "An introduction to the Vodafone GPRS environment and supported services," Vodafone Ltd., Dec. 2000.

[7] "The VCDiff generic differencing and compression data format," IETF, draft-korn-vcdiff-06.txt , Nov. 2001.

[8] Y. Shapira, "NetGain—mobile data access platform," in *Proc. Workshop Internet Usage Over 2.5G and 3G, IST-2001-92 125*, Barcelona, Spain, Mar. 2003, [Online]. Available: http://www.flashnetworks.com.

[9] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. IEEE ICDCS*, May 1995, pp. 136–143.

[10] A. Fox, S. Gribble, Y. Chawathe, and E. Brewer, "Adapting to network and client variation using active proxies: Lessons and perspective," *IEEE Pers. Commun.*, vol. 5, no. 4, pp. 10–19, Aug. 1999.

[11] A. Gurtov, M. Passoja, O. Aalto, and M. Raitola, "Multi-layer protocol tracing in a GPRS network," in *Proc. IEEE Fall VTC*, Sep. 2002, pp. 1612–1616.

[12] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proc. IEEE INFOCOM*, 2001, pp. 1801–1810.

[13] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proc. ACM Sigcomm Internet Measurement Workshop (IMW)*, 2001, pp. 169–182.

[14] B. C. Housel and D. B. Lindquist, "WebExpress: a system for optimizing web browsing in a wireless environment," in *Proc. ACM Mobicom*, 1996, pp. 108–116.

[15] GPRS link characterization. [Online]. Available: http://www.cl.cam.ac.uk/users/rc277/linkchar.html

[16] G. Barish and K. Obraczka, "World wide web caching: Trends and techniques," *IEEE Commun. Mag.*, vol. 38, no. 5, pp. 178–184, May 2000.

[17] G. Brasche and B. Walke, "Concepts, services and protocols of the new GSM phase 2+ general packet radio service," *IEEE Commun. Mag.*, pp. 94–104, Aug. 1997.

[18] H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless web performance," in *Proc. IEEE GLOBECOM*, 1998.

[19] H. Balakrishnan, R. H. Katz, and S. Seshan, "Improving TCP/IP performance over wireless networks," in *Proc. ACM Mobicom*, 1995, pp. 2–11.

[20] H. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilly, "Network performance effects of HTTP/1.1 CSS1, and PNG," in *Proc. ACM Sigcomm*, 1997, pp. 155–166.

[21] J. C. Mogul, "Support for out-of-order responses in HTTP," Internet Draft, Network Working Group, Apr. 2001.

[22] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, "Potential benefits of delta encoding and data compression for HTTP," in *Proc. ACM Sigcomm*, 1997, pp. 181–194.

[23] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Mar. 2000.

[24] M. C. Chan and R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation," in *Proc. ACM Mobicom*, 2002, pp. 71–82.

[25] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K. Raatikainen, "Optimizing world-wide web for weakly-connected mobile workstations: An indirect approach," in *Proc. Workshop Serv. Distrib. Networked Environments (SDNE)*, 1995, pp. 132–139.

[26] M. Meyer, "TCP performance over GPRS," in *Proc. IEEE WCNC*, 2000, pp. 1248–1252.

[27] P. Rodriguez and V. Fridman, "Performance of PEP in cellular wireless networks," in *Proc. 8th Int. Workshop Web Content Caching Distrib.*, Sep. 2003.

[28] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," in *Proc. ACM Mobicom*, 1999, pp. 231–241.

[29] V. N. Padmanabhan and J. C. Mogul, "Improving http latency," *Comput. Netw. ISDN Syst.*, vol. 28, no. 1, pp. 25–35, Dec. 1995.

[30] R. Braden, "T/TCP—TCP extensions for transactions," IETF, Request for Comments, RFC 1644, 1994.

[31] R. Chakravorty and I. Pratt, "Performance issues with general packet radio service (GPRS)," *J. Commun. Netw. (JCN)—Special Issue on Evolving From 3G Deployment to 4G Definition*, vol. 4, no. 3, pp. 4–19, Dec. 2002.

[32] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt, "Performance optimizations for wireless wide-area networks: Comparative study and experimental evaluation," in *Proc. ACM Mobicom*, 2004, pp. 159–173.

[33] R. Chakravorty, S. Katti, I. Pratt, and J. Crowcroft, "Flow aggregation for enhanced TCP over wide-area wireless," in *Proc. IEEE INFOCOM*, 2003, pp. 1754–1764.

[34] R. Fielding *et al.*, "Hypertext transfer protocol—HTTP/1.1," IETF, Request For Comments, RFC 2616, 1999.

[35] R. Kalden, I. Meirick, and M. Meyer, "Wireless Internet access based on GPRS," *IEEE Pers. Commun.*, pp. 8–18, Apr. 2000.

[36] R. Ludwig, B. Rathonyi, A. Konrad, K. Oden, and A. Joseph, "Multi-layer tracing of TCP over a reliable wireless link," in *Proc. ACM Sigmetrics*, 1999, pp. 144–154.

[37] S. Floyd and L. Diagle, "IAB architectural and policy considerations for open pluggable edge services," IETF, Request for Comments, RFC 3238, Jan. 2002.

[38] T. B. Fleming, S. F. Midkiff, and N. J. Davis, "Improving the performance of the world wide web over wireless networks," in *Proc. GLOBECOM*, 1997, pp. 1937–1942.

[39] T. Go, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments," in *Proc. IEEE INFOCOM*, 2000, pp. 1537–1545.

[40] T. Kelly and J. Mogul, "Aliasing on the world wide web: Prevalence and performance implications," in *Proc. World Wide Web (WWW) Conf.*, 2002, pp. 281–292.