Performance Issues with General Packet Radio Service

Rajiv Chakravorty and Ian Pratt

Abstract: **The General Packet Radio Service (GPRS) is being deployed by GSM network operators world-wide, and promises to provide users with "always-on" data access at bandwidths comparable to that of conventional fixed-wire telephone modems. However, 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 examine the causes, and show how unfortunate interactions between the GPRS link characteristics and TCP/IP protocols lead to poor performance. A performance characterisation of the GPRS link-layer is presented, determined through extensive measurements taken over production networks. We present measurements of packet loss rates, bandwidth availability, link stability, and round-trip time.

The effect these characteristics have on TCP behaviour are examined, demonstrating how they can result in poor link utilization, excessive packet queueing, and slow recovery from packet losses. Further, we show that the HTTP protocol can compound these issues, leading to dire WWW performance. We go on to show how the use of a transparent proxy interposed near the wired-wireless border can be used to alleviate many of these performance issues without requiring changes to either client or server end systems.

Index Terms: **GPRS, Wireless, HTTP, TCP, Proxy**

I. Introduction

Throughout the world, GSM cellular mobile networks are being upgraded to support the General Packet Radio Service (GPRS). GPRS offers an "always on" connectivity to mobile users, with wide-area geographical 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 TCP and HTTP performance over GPRS. There are some interesting simulation studies [1], [4], but we have found actual deployed network performance to be somewhat different.

Practical performance issues observed over GPRS can be shared to some extent with wireless LANs like 802.11b [21], satellite systems, and other wide-area wireless schemes such as Metricom Ricochet [24] and Cellular Digital Packet Data (CDPD) [33]. However, GPRS presents a particularly challenging environment for achieving good application performance. Take for instance TCP - The Internet's *de-facto* transport protocol tuned to detect congestion and avoid overload. Unfortunately, TCP performance degrades over wireless links where losses are mostly non-congestive, predominantly due to external environmental factors such as fading, interference etc. Our link characterization measurements reveal that GPRS links have very high RTTs (>1000ms), fluctuating bandwidths along with occasional link outages. Hence TCP performance suffers in several ways:

- A slow-start phase that takes many seconds (due to high RTTs) for the window to ramp-up and fully utilize the link,
- Excess queuing over the downlink that can result in gross unfairness to other TCP flows, and a high probability of timeouts during intial connection request,
- Spurious TCP timeouts due to occasional link 'stalls',
- Slow recovery (many seconds) after timeouts.

At least theoritically, GPRS can achieve a much higher data rate (>100kb/s) when compared to a single GSM circuit switched line (9.6kb/s). This makes it particularly suitable for bursty applications like the World Wide Web (WWW). However, WWW (or HTTP's) dependence on TCP presents a set of pressing performance issues. As we show how large HTTP transfers can lead to excess queueing over the downlink. This can harm other existing, or new flows, with the potential to cause gross unfairness.

Web browser behaviour also has a substantial effect on page download times over GPRS. In an effort to improve response times on wired-Internet links, client browsers open several concurrent TCP connections. We show that such behaviour on the part of the web clients may result in saturation of the downlink buffers, and increased control overhead that can negatively impact page download times over GPRS. We attempt to answer questions like:

- *What is the "typical" GPRS link behaviour, and how does TCP peform over GPRS?*
- *How fair is TCP over GPRS? Can the unfairness affect Web performace?*
- *How does browser behaviour influence page download times?*
- *What is the quantitative benefit achievable when requests are pipelined?*

To answer these questions, we performed a number of experiments over Vodafone UK's GPRS infrastructure. To support the underpinnings of the experiments, we conducted similar, but less thorough performance measurements on several other European GPRS networks, and found performance to be very similar. From analysis of experiments conducted over our test bed, we have determined why TCP and HTTP can under-perform over GPRS. We

The authors are with University of Cambridge Computer Laboratory, William Gates Building, Cambridge CB3 0FD, England. e-mail: {rajiv.chakravorty, ian.pratt}@cl.cam.ac.uk

1229-2370/99/$10.00 © 2002 KICS

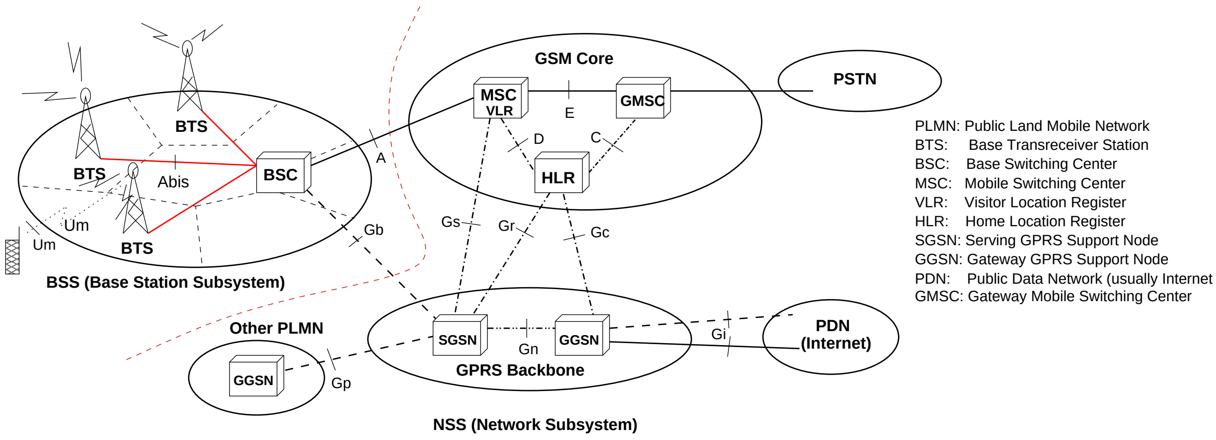

Fig. 1. The GSM-GPRS network

then introduce a number of simple yet effective ways to overcome many of the problems. These are implemented in an interposed 'transparent proxy' that improves performance without requiring changes to either client or server end-systems. We demonstrate that significant performance improvements are possible with such an approach.

The paper is structured as follows: The next section provides a brief overview on GPRS. Section III presents our GPRS link characterization experiements. Section IV examines HTTP/TCP problems over GPRS. In section V, we describe our transparent proxy-based scheme to improve protocol performance, and demonstrate its effectiveness. Section VI discusses related work, while the final section summarises our findings and suggests directions for future research.

II. The GPRS Overview

Shown in figure 1 is a GSM network with GPRS extensions. As part of the transition towards GPRS, new components have been added in the network subsytem (NSS) to the traditional GSM network. The two new nodes - SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Suport Node) are used for GPRS that will be later upgraded for third generation Universal Mobile Telecommunication Network (3G-UMTS). The SGSN node acts as a packet switch that performs signalling similar to a mobile switching center (MSC) in GSM, along with cell selection, routing and handovers between different Base Switching Centers (BSCs). It controls the mobile terminal's access to the GPRS network and routes packets to the appropriate BSC. When migrating to UMTS, SGSN will be enhanced to replace the MSC altogether, where it will switch packets to the correct UMTS terrestrial radio network (UTRAN) [7]. The GGSN is the last "port of call" that acts as a gateway between the mobile packet routing of GPRS, and the IP routing of the Internet. The MSC/visitor location register (VLR), Home Location Register (HLR), and short message service (SMS) center are functional entities tied to the circuit-switched GSM. To exchange GPRS subscriber information with the SGSN, the HLR is extended by a GPRS register (GR).

A mobile terminal (MT) wishing to use GPRS will first *attach* itself to the network through a signalling 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 MT device class, it can connect to either 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.

A reliable RLC (radio link control) mode ensures that packets are delivered in order, while a *selective repeat* ARQ (automatic repeat request) coupled with the modulo-128 numbering of data blocks using temporary RLC flow identifiers (TFI) helps to recover from packets received in error. In this scheme, the sender transmits blocks within a window of 64 blocks, and the receiver side periodically sends ACK/NACK messages. While every ACK acknowledges all correctly received RLC blocks indicated upto a block sequence number (BSN), the NACKs act as a bitmap to selectively request erroneously received RLC data blocks for re-transmission [4]. The sender then just re-transmits the erroneous RLC data blocks advancing the sending window. However, all this happens at the expense of variable throughput and higher delay due to retransmissions [15].

Radio conditions change with time, and achievable data rates over GPRS can vary, depending on other factors, primarily on external environmental interference. Higher interference will lead to higher block error rates over GPRS, and consequently, longer data transfer times. The level of interference is typically specified in channel-to-interference (C/I) ratio of the radio environment. A low C/I (for e.g. $<$ 6-8dB) gives tough radio conditions (high block error rates), a C/I of 13-18dB indicates moderate radio conditions while a high C/I (e.g.$>$ 25dB) gives good channel conditions. GPRS copes with a wide range of radio/channel conditions by making use of 4 different coding schemes (CS-1 TO CS-4) [1][3] with varying levels of FEC (forward error correction). Most of the currently deployed GPRS networks support only CS-1 and CS-2 [40] – the other two are not used as block error rates would be typically too high for the applications to be useful. While CS-1 is meant for use during tough radio conditions (e.g. < 7-8dB of C/I), CS-2 is useful during tough-to-moderate channel conditions (e.g. < 15-18dB of C/I). GPRS network operators usually determine that CS-1 and CS-2 are a good compromise coding scheme for the moment. Later, when applications become more error resilient then it should be possible to use even CS-3 and CS-4. An other difficulty using CS-3 and CS-4 is that it cannot be supported by many GPRS networks, since the '`Abis`' and '`Gb`' interface (see figure 1) is currently capacity limited. The CS-4 scheme removes FEC correcting capabilities altogether, while CS-2 scheme employs a coding rate of approximately 2:3, to obtains a transmission rate as high as 13.4 kbit/s per GSM time slot [3]. The effective GPRS data rate is slightly less, due to protocol header overhead and signalling messages.

Radio resources of a cell are shared between all GPRS and GSM mobile stations located in the 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* [3]) 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.

When there is contention for GPRS resources, individual PDCHs may be multiplexed between different users. When this occurs, the specification allows for packets to be prioritised according to various Quality of Service (QoS) levels. A user can request for a desired QoS profile during the packet data protocol (PDP) context activation phase.

GPRS Release 99 defines several QoS parameters to meet the application requirements for different levels of network QoS. The release offers several benefits when compared to its predecessor (Release 97/98) [5] - such as - BSS aware QoS profile negotiation, MT and GGSN initiated QoS profile (re)negotiation based on application or network requirements, and multiple PDP contexts per PDP address. Further, four distinct GPRS traffic classes are specified: *conversational*, *streaming*, *interactive* and *background* [1], [5]. Applications that are delay sensitive belong to the *conversational* class. The *conversational* class offers strict delay and bandwidth gaurantees, while the *background* class offers no quantitative or qualitative guarantees. It can be at best referred to as the *best-effort* traffic class. Currently, in the 'phase one' of GPRS deployment, operators only support a single *best-effort* service class [40]. Further information about GRPS network design and operation can be found in [1-6].

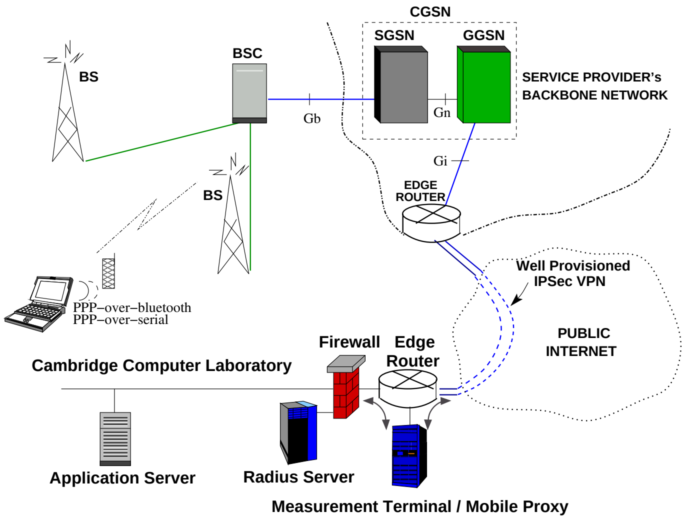

Fig. 2. Test Bed Set-Up for Link Characterization

III. GPRS Network Characterization

A. Test Environment and Tools

Our experimental setup for characterizing GPRS links is shown in figure 2. The measurements presented in this paper were all performed over Vodafone UK's national GPRS network, though we have recorded similar results on a number of other European GPRS networks. A number of different handsets from different vendors have been tried; we found no significant performance variation between handsets of the same GPRS device class. Measurements have been repeated at many different locations (typically in isolated cells) resulting in a wide range of radio and network conditions to try and gain a view of 'typical' performance experienced by a user.

In an attempt to determine whether GSM time slot contention was common, we repeated measurements at different times of the day at locations we suspected of being 'busy'. However, throughout our tests we saw little evidence of network contention occurring. This is perhaps to be expected due to the currently small number of GPRS users and the generous time slot provisioning employed by Vodafone.

In an example test set-up (figure 2), a laptop connects to a Motorola T260 GPRS (3+1) (3 downlink, 1 uplink channels) phone through a serial PPP (point-to-point) link to act as a GPRS mobile terminal. Adhering to the usual GPRS architecture, the base stations (BSs) are linked to the SGSN which is then connected to a GGSN. In the current Vodafone configuration, both SGSN and GGSN node are co-located in a CGSN (Combined GPRS Support Node) [40]. The test network used logical link control (LLC) layer in an unacknowledged mode with protocol header compression turned-off. Turning off header compression alleviates the extra computation overhead at the core network, and avoids additional performance problems arising from packet losses [38].

Since we were unable to install equipment next to the

CGSN we made use of a well provisioned IPSec (IP Security) VPN tunnel to route all traffic via the Computer Laboratory. The measurement terminal (see figure 2) was then located at the end of the tunnel, with routing configured so that all packets flowing to and from the mobile host are passed to it for processing. A RADIUS server was used to authenticate mobile terminals and assign IP addresses.

All the characterization tests were performed using a version of the `ttcp` program modified (`ttcp+` [41]) to enable traffic streams to be generated at specified rates and with particular burst characteristics. We also inserted time stamps and sequence numbers in packets to track time-in-flight between sender and receiver (using NTP synchronized clocks) and to detect packet loss and re-ordering.

B. Results from Link Characterization

Tests were performed to measure up and downlink packet latencies for different packet sizes, and up/downlink bandwidth (both TCP and raw bandwidth). During all these tests, any incidence of packet loss or re-ordering was noted. In all cases, the mobile terminal was stationary, though a number of locations were used during the bandwidth measurements presented later.

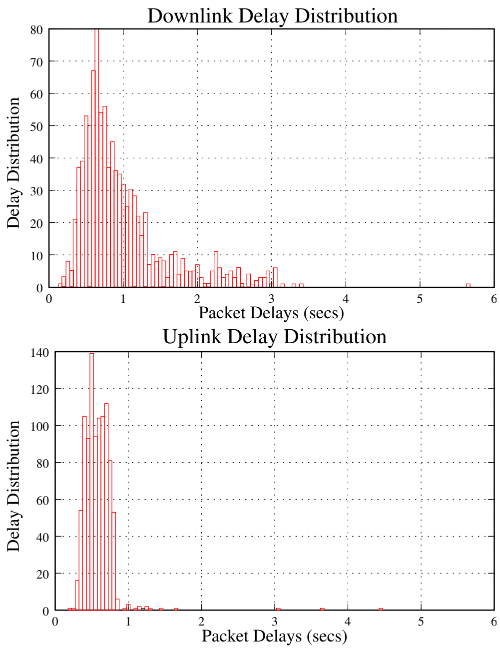

Fig. 3. Single packet time-in-flight delay distribution plots showing (top-down) (a) downlink delay (b) uplink delay distribution. Measurements involved transfer of 1000 packets with random intervals > 4s between successive packet transfers.

The tests revealed that GPRS, like many other wide-area wireless networks, exhibits many of the following characteristics: low and fluctuating bandwidth, high and variable latency, and occasional link 'blackouts' [18], [19]. A comprehensive report on GPRS link characterization is available in the form of a separate technical report [37]. We discuss some key findings:

High and Variable Latency:- Figure 3 presents a histograms of time-in-flight latency for one thousand 64 byte UDP datagrams sent with a random spacing of between 5 and 10 seconds during good radio conditions. The RTT experienced by a connection is equal to the sum of the uplink and downlink distributions. As can be observed, latencies are large and highly variable, particularly in the downlink direction. RTTs of around a second are commonplace, making the service poor for interactive applications.

If the experiment was repeated by sending bursts of several packets, it can be observed that it is only the first packet in a burst that experiences the high jitter: following packets tend to arrive with quite a tight jitter bound, unless there is evidence that the packet was retransmitted due to loss signalled by ARQ. This indicates that a substantial proportion of the latency is incurred when the link to a mobile terminal transitions from previously being idle. 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 flow (TBF) [6], [38]. 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. The latest release (GPRS 1999) does consider an extended TBF life-time; however, this optimization could lead to inefficient scheduling at the base station controller (BSC), with only some improvement ($\sim$ 100msec) in overall RTTs [38].

Fluctuating Bandwidth:- We observe that signal quality leads to significant (often sudden) variations in bandwidth perceived 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. A number of other factors contribute to throughput values lower than the maximum possible, for more description here, see [4].

Raw UDP bandwidth was measured using `ttcp` [41] to send a continuous stream of 1024 byte packets at a rate just above what the radio link is capable of carrying. Bandwidth measurements taken at the receiver averaged over fixed intervals enable variations in raw link bandwidth to be observed (packet losses in this experiment were ignored as they are most likely due to packet discard at the CGSN).

As discussed in section II, the achievable data throughput over GPRS will critically depend on the carrier-to-interference ratio (C/I) of the radio environment [4]. Depending on the C/I of the radio environment and corresponding coding scheme used, will result in different block

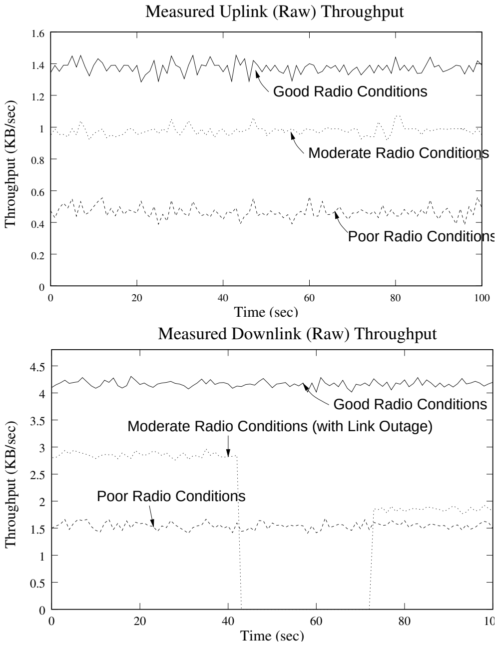

Fig. 4. Raw throughput measurements using Mototola (3+1) GPRS phone for the case of (top-down) (a) uplink and (b) downlink.

error rates that will consequently impact application data throughput. Figure 4 shows raw UDP bandwidth traces for the uplink and downlink directions taken under a number of different radio (C/I) conditions. Note how the available bandwidth often varies with time with changing radio conditions (we believe there was no contention for time slots).

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 in stationary conditions. However, packets can still be lost, mainly for two reasons: (1) deep fading (and/or interference, shadowing etc.) leading to bursty *radio losses* that persist for longer than the link-layer is prepared to keep retransmitting a packet, and (2) during cell reselections due to cell update procedure (or even routing area update) that can lead to a link *black-out* from few to several seconds. In both cases, consecutive packets in a window are usually lost. Most current GPRS mobile terminals (based on GPRS 97 release) perform cell-reselections based on signal-strength. Thus in overlapping-cells, cell-reselections may happen even during mobile-hosts' stationary conditions. Unfortunately, this can have implications on application performance, hence *network-assisted* cell updates are being considered [38].

Link Outages:- Shown in figure 4, a case when bandwidth available on the downlink channel drops to zero for a period of 30 seconds in the middle of one of the traces. Unfortunately, such link 'blackouts' are not uncommon, particularly when the mobile terminal is on the move in a car or train and during cell-handoffs. Link outages that were observed, typically last for 5-30s. However, due to the RLC's ARQ protocol packets are rarely lost, just grossly delayed.

We have noticed outages during stationary conditions, though the outage interval in this case is small. Sudden signal quality degradation, prolonged fades and intra-zone handovers (cell-reselections) can lead to such link black-outs. When link outages are of small duration, packets are just delayed and are lost only in few cases. In contrast, when outages are of higher duration there tend to be burst losses.

Additionally, we also observed downlink transfers to stop altogether during transfers. We believe this to be a specific case of link-reset, where a mobile terminal would stall and stop listening to its TBF. We currently believe this to be due to inconsistent timer implementations within mobile terminals and base station subsystem (BSS). In such cases, it is necessary to terminate and restart the point-to-point (PPP) session.

IV. TCP Performance over GPRS

As well as the link characterization measurements, we also performed separate tests to gain a better insight into TCP performance over GPRS, specifically over the downlink channels. We targeted the downlink channel because of its importance in current mobile applications such as web browsing, reading email etc.

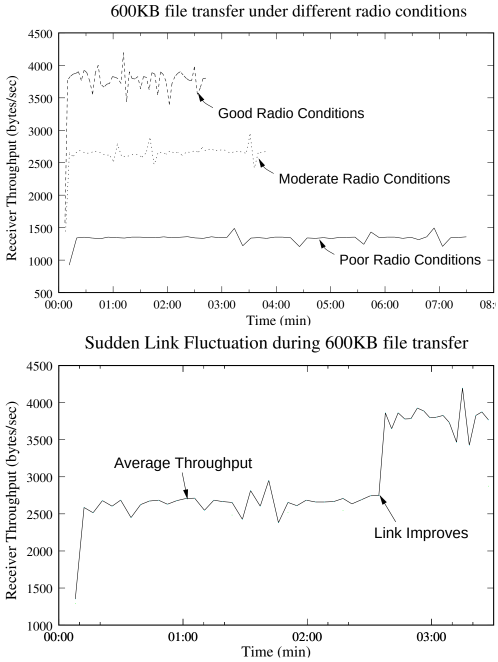

Fig. 5. Plot (top-down) (a) shows the 600KB transfer progressing under three different radio conditions. Plot (b) shows how bandwidth can change during the course of a transfer

In this experiment, file transfer tests were performed during different radio conditions, and traces of the transfer collected using `tcpdump` [41] run at both ends: the sending host in the Lab, and the mobile receiver laptop connected via a GPRS phone. Both hosts used Linux version 2.4, which employs a modern TCP implementation supporting Selective ACKnowledgements (SACKs) [9].

The traces were analysed using `tcptrace` [41]. To under-

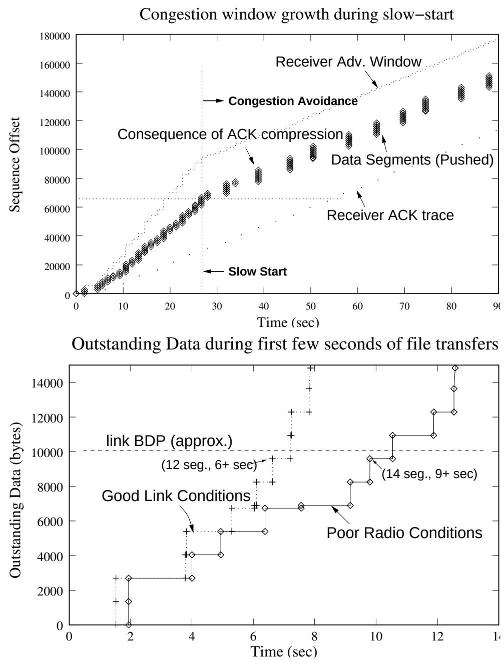

Fig. 6. Plot (top-down) (a) shows the characteristic exponential congestion window growth due to slow-start (SS). Plot (b) shows outstanding data in case of timeout due to a dupack(sack).

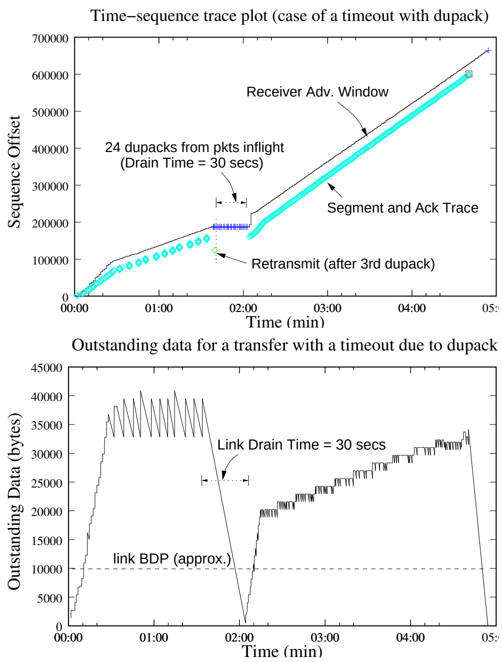

Fig. 7. Case of timeout due to a dupack(sack). Plot (top-down) (a) shows the sender sequence trace and plot (b) shows corresponding outstanding data.

stand steady-state link behaviour, we selected a reasonably large file transfer size of 600KB. Figure 5(a) shows throughput measured at the receiver averaged over 10 packets for three different file transfer runs performed under different radio conditions. As can be seen, there are wide variations in throughput and hence download completion time. In figure 5(b) we observe a sudden improvement in available link bandwidth. In this case, there are sufficient packets already queued at the GPRS router that the TCP connection is able to seamlessly utilize the extra bandwidth without having to grow the congestion window further.

In the following sections, we look at more detailed traces to describe some of the specific performance issues observed during TCP transfers over GPRS.

A. TCP Start-up Performance

Figure 6 (b) shows a close up of the first few seconds of the connection, alongside another connection under slightly worse radio conditions. An estimate of the link bandwidth delay product (BDP) is also marked, approximately 10KB. 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. For a TCP connection to fully utilize the 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 6 seconds to ramp the congestion window up to the link BDP, measured from when the intial connect request (TCP's SYN) was made. Hence, for transfers shorter than about 18KB, TCP fails to exploit even the meagre bandwidth that GPRS makes available to it. Since many HTTP objects are smaller than this, the effect on web browsing performance can be dire.

A further point to note about figure 6(a) 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, hence it is surmised that the ack compression [25] occurs as a result of the GPRS uplink (since the wired network is well provisioned). 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. The 'bunching' on the uplink can be due to the GPRS link layer. In GPRS the size of a single LLC frame structure may vary from a minimum of 140 bytes to a maximum of 1520 bytes. The RLC layer then operates on small blocks of user data (usually 20-50 bytes, depending upon coding scheme in use [38]) before it is finally transmitted. Thus even if one minimum size LLC frame is buffered for transmission, it can generate at least 3 back-to-back TCP ACKs resulting in bunching.

B. Excess Queuing

Due to its low bandwidth, the GPRS link is the bottleneck link in most connections, and so packets destined for the downlink get queued up at the GPRS CGSN. We found that the existing GPRS infrastructure offers substantial buffering: UDP burst tests indicate over 120KB of buffering. Therefore for a long enough session, TCP's congestion control algorithm could fill the entire router buffer before incurring packet loss and reducing its window. In practise, the window is typically not allowed to become quite so excessive due to the receiver's flow control window, which in most TCP implementation is limited to under 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 queuing delay. Figure 7 shows a TCP connection in such a state, where there is 40KB of outstanding data leading to a measured RTT of around 30 seconds. Excess queuing complicates a number of issues:

- **RTT Inflation**:- Higher queuing delays can severely degrade TCP performance [20]. A second TCP connection established over the same link is likely to have its initial connection request timed-out [15].
- **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 [15].
- **Problems of Leftover (Stale) Data**:- For downlink channels, the data in the pipe may become obsolete when a user aborts a web download and abnormally terminates the connection. Draining leftover data from such a link may take on the order of several seconds.
- **Higher Recovery Time**:- Recovery time from timeouts due to dupacks (or sacks) or coarse timeouts in TCP over a saturated GPRS link is high.

C. Recovery over GPRS

Shown in figure 7 is the TCP's performance during a packet loss due to dupack (in this case SACKs). The point to note is the large time (30 seconds) 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 the use SACKs in the GPRS environment: without SACKs 40KB of correctly received data would have been discarded.

D. Fairness between flows

Excess queuing can lead to gross unfairness between competing flows. Figure 8 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 queuing 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 [27]. 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.

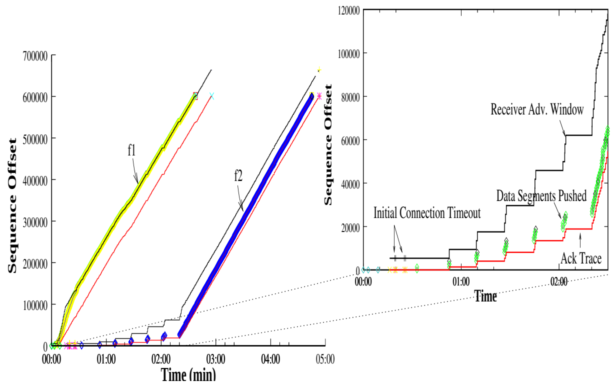

Fig. 8. Close-up of time sequence plots for two concurrent file transfers over GPRS, where `f2` was initiated 10 seconds after `f1` .

V. Web Client Performance

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. Typically, by opening more connections, browsers exhibit greater aggressiveness to get a higher share of the link bandwdith. Also, with more connections, browsers implicitly avoid *head-of-line* (HOL) [12] blocking problems. An aggressive browser will obviously reap benefits over conventional high bandwidth links shared by many users. In contrast, the approach can be deleterious over GPRS links.

Using multiple connections over 'long-thin' GPRS links has a number of drawbacks: 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+LLC=55 bytes, as in [40]) even for data packets that are exchanged over the link. Second, every TCP connection is associated with a transaction or connection set-up (3-way TCP handshake) overhead. The delay this introduces is very 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 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.

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 [28]. Use of HTTP/1.1 persistent connections, where the browser and server employ the same TCP connection for transfer of multiple objects is gradually becoming more widespread. However, use of HTTP/1.1 pipelined-persistent connections (where multiple outstanding requests are permitted on the same connection) is currently almost non existent. The use of persistent and pipelined connections can eliminate a substantial amount of control and connection setup overhead, thereby resulting in higher utilization over GPRS. In the following sections, we evaluate the pipelining benefits analytically and in a section later validate the benefits that can be achieved through experiments conducted over the GPRS test-bed.

A. Pipelining Incentives over GPRS

Persistent connections allow multiple requests to be issued on the same TCP connection. However, a new request can only be issued after receiving a complete response from the server. HTTP/1.1 *pipelined* connections allow multiple requests to be kept outstanding before a response is received. Thus, the server is able to overlap processing of requests. Often, this means that the server is able to maintain the congestion window when transferring data to the client, avoiding the need to re-enter slow-start between each object as is required for non-pipelined connections. Previous studies have demonstrated substantial improvement in page download times using pipelined connections [10], [11]. However, none of the earlier literatures appear to quantify benefits associated with the degree of pipelining *i.e.* pipelining effectiveness. If we assume that a web client could pipeline its requests as aggressively as possible over a given connection, then how do we measure pipelining effectiveness? We introduce a new term *Pipeline Factor* (Φ). Pipeline factor typically represents pipelining aggressiveness of a web-client. A high value of 'Φ' indicates that a web client is able to keep more requests outstanding during a connection's lifetime.

We define Φ for a connection 'i' as:

$$\Phi_i = \frac{\sum_{z=1}^{n} \beta_z(t)}{n} \tag{1}$$

where n corresponds to the total number of requests made on a connection. Here $\beta_z(t)$ is the *pipeline index* calculated separately for each request as the total number of requests (including the request 'z') minus the responses received before request 'z' was made. The maximum possible Φ for a connection is ($\Phi_{max}=\frac{\sum_{z=1}^{n} z}{n}$), which happens when all the requests are pipelined before a response is received. Pipeline factor Φ can achieve a minimum value of 1, when a connection is effectively persistent with no pipelining; there is only ever one request outstanding during the connection lifetime. Notice equation 1 gives higher weight to connections that can keep more requests outstanding. Having more requests outstanding (high Φ) means a server (or proxy) is more likely to keep the downlink busy and thus achieve high link utilization. Using a reduced number of concurrent connections (to minimize protocol overhead) and making aggressive use of pipelining is of particular benefit on high latency links such as GPRS. We revisit this topic, when we discuss our experimental test results later.

Figure 9 shows a sample Φ calculation for a pipelined connection. As shown, the pipeline index for the the 3^{rd} and the 4^{th} request is the same for both: 2 each. The index value for the 4^{th} request is 2 as its index gain is negated by the reception of two responses (obtained as, 4 requests - 2 responses = 2). The Φ value comes out to about 1.8 from a maximum possible Φ_{max} of 3 for this connection.

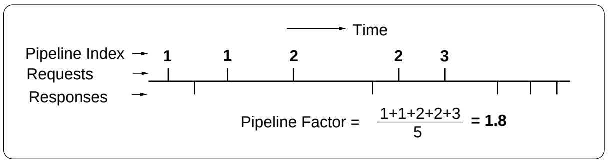

Fig. 9. Pipeline factor (Φ) calculation of a pipelined connection

B. Pipelining Efficiency

We introduce another term *pipelining efficiency* (η). Browsers can in principle pipeline requests on more than one connection – the pipelining efficiency in such a case will be determined by how effectively requests are pipelined across all connections. The overall pipelining efficiency for 'm' connections can be given by:

$$\eta = \frac{\sum_{i=1}^{m} \Phi_i}{\sum_{i=1}^{m} (\Phi_i^{max})} = \frac{\sum_{i=1}^{m} \Phi_i}{\sum_{i=1}^{m} \left(\frac{\sum_{z=1}^{n} z}{n} \right)} \tag{2}$$

It is typically not possible to achieve 100% pipelining efficiency (η), as they would then have to know the URLs of all objects that make up a page beforehand. Browsers typically need at least one response (for non-frame based static web-pages) to parse and make subsequent requests for other *inlined* objects. For web sites offering dynamic content, strict resource dependencies can further restrict the level of pipelining, which lowers 'Φ' over a given connection. The simplest is for a web-site offering static content, where after receiving the first response, all other requests could be immediately pipelined. Using experiments conducted over GPRS test bed, we discuss how 'Φ' for a web client can effect web download performance.

The majority of current web browsers have yet to offer any support for pipelining. Notable exceptions are mozilla [44] and opera [45] where support for pipelining exists, but the option needs to be explicitly enabled by the user. However, it is apparent that neither browser currently makes particularly aggressive use of pipelining, missing out on many potential occasions where it could be exploited.

VI. Improving Performance over GPRS

In previous sections, we identified causes of poor performance over GPRS links, and here we report on our efforts to make improvements. One fundamental constraint was that we wished to improve performance without requiring end-system software modifications: Experience shows that the vast majority of schemes that require such end-system changes never see widespread deployment.

Our scheme makes use of a modified HTTP proxy and TCP protocol stack running on a machine located near the wired-wireless border. Users may configure their browsers to explicitly use the proxy, or transparent proxying [16] may be employed, as is done by many fixed-wire Internet

Service providers today. The objectives of our proxy are as follows:

Better Utilization – A higher utilization over the wireless link, especially downlink, to benefit the usually 'short' web flows.

Faster Recovery – We wanted our proxy to quickly recover from occasional link stalls during hand-offs and during occurrence of burst losses.

Minimize Queuing – One important goal in the proxy was to alleviate the problem of excessive queuing at the wired-wireless gateway.

Restore Fairness – Allocate bandwidth fairly irrespective of the length of the flows.

To achieve these goals, we provide performance enhancements in the proxy at two different layers - transport layer (TCP) and application layer (HTTP). We refer to these proxy enhancements independently as TL-E for transport level enhancement, and AL-E to signify application level enhancement. We discuss TL-E and AL-E further.

A. Transport Level Enhancement (TL-E)

A major cause of poor performance with TCP over GPRS is link under utilization during the first 6-10 seconds of a connection due to the pessimistic nature of the slow start algorithm and high RTTs of the GPRS links.

Slow start is certainly an appropriate mechanism for the Internet in general, but within the proxy, information is available with which we can make a better decision about the congestion window size.

Hence a substantial benefit can be offered to typically short and bursty web sessions by avoiding slow-start and instead making use of the full capacity of the downlink during connection startup. In order to achieve this, our proxy uses a modified TCP sender over the wireless link that uses a fixed size congestion window (but remains compatible with unmodified receivers). The size is fixed to a relatively static estimate of the Bandwidth Delay Product (BDP) of the link. Thus, in the downlink direction, slow start is eliminated and further unnecessary growth of the congestion window beyond the BDP is avoided. We call this TCP `cwnd` clamping.

Achieving the same optimisation in the uplink direction can be achieved by re-writing the receive flow-control window of returning ACKs. However, due to the asymmetric nature of most web transfers such optimisation is generally not worthwhile.

The underlying GPRS network ensures that bandwidth is shared fairly amongst different users, and hence there is no need for TCP to be trying to do the same based on less accurate information. Ideally, the CGSN could provide feedback to the proxy about current radio conditions and time slot contention, enabling it to adjust the 'fixed' size congestion window, but in practice this is currently unnecessary.

Once the proxy is successful in sending C_{clamp} of data it goes into a self-clocking state in which one segment (from whatever connection the scheduler has selected) is released each time an ACK for an equivalent amount of data from the receiver. With an ideal value of C_{clamp}, the link should never be under utilised if there is data to send, and there should only ever be minimal queueing at the CGSN gateway. Typically, we find the optimal C_{clamp} value to be 10-15% higher than the that calculated by multiplying the maximum link bandwidth by the typical link RTT. This excess is required due to link jitter, use of delayed ACKs by the TCP receiver in the mobile host, and ACK compression occurring due to the link layer.

The `cwnd` remains clamped even during times of poor link performance *i.e.* during handoff's, interference or fading. While starting with a fixed value of `cwnd`, the mobile proxy needs to ensure that any initial packet burst does not overrun link buffers. Since the bandwidth-delay product (BDP) of current GPRS links is small (e.g. ≈10KB), this is not a significant problem at this time. For future GPRS devices supporting more downlink channels, the proxy may need to use traffic shaping to smooth the initial burst of packets to a conservative estimate of the link bandwidth.

In the case of a packet loss, we preserve the `cwnd` value, clocking out further packets only when ACKs are received. RTO triggered retransmissions operate in the normal manner.

A.1 Validating Transport Level Enhancements (TL-E)

In this section, we evaluate the efficacy of TL-E. The TL-E in the proxy offers the following benefits:

- **Reduced Queuing Delays**:- Excessive queuing is reduced by limiting TCP data over the link. As a consequence, RTT inflation and its impact on retransmit timer values are also minimized.
- **Faster Startup**:- It avoids slow-start and instead makes full use of the downlink capacity. This improves start-up performance of short connections and reduces overall transfer times.
- **Quick Recovery from Losses**:- TCP *cwnd clamping* reduces drain time during losses leading to quick TCP recovery. By limiting data over the link, spurious retransmission cycles due to sudden delay fluctuations can be avoided. This also reconciles with other negative effects such as stale (or leftover) TCP data due to abnormal disconnections.

1. Minimizing Excess Queuing using TCP clamp:- We conducted a series of file download tests over GPRS, with and without the presence of the mobile proxy implementing our clamping strategy. For these tests we used fixed values of the clamped window (C_{clamp}).

Transfer tests were performed with an initial value of 4KB and we increased this to 32KB in a number of steps. Figure 10(a) shows typical traces for the 600KB file transfers corresponding to different values of `cwnd`. It is evident that the transfer times for all the runs except when `cwnd` = 4KB run (a case of link under utilization) are almost same. A `cwnd` value of 10KB (corresponding to 9.5KB when integer numbers of segments are considered) or higher ensures the link is fully utilized. An 8KB window typically yields similar results, though we have observed circumstances in

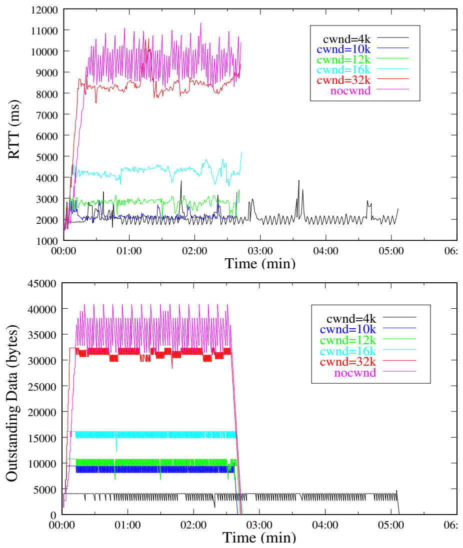

Fig. 10. Figures showing (top-down) (a) sender perceived RTTs and (b) Outstanding (inflight) TCP data during 600KB file transfer. Effects of queuing delays can be effectively reduced by clamping the congestion window. A good selection of C_{clamp} ensures that link is never underutilized. A choice of cwnd=10KB (reflecting BDP of the link) was found to be appropriate for our GPRS mobile terminal.

which under utilization has occurred due to ACK compression; for clarity the line is omitted.

Higher values of `cwnd` leads to higher values of resultant RTTs. Using a `cwnd` of 10KB results in a low and relatively stable RTT, similar to the 4KB case. Other values progressively tend toward the large and very variable RTT incurred in the absence of the mobile proxy.

2. Benefitting from slow-start elimination:- To quantify the benefits of avoiding slow-start for short TCP sessions, we used `ttcp` to perform a series of short (5KB-30KB) downloads, primarily to reflect web sessions behaviour. Each transfer for a given size was repeated 25 times and traces recorded using `tcpdump`.

Figure 11 shows that transfer times for TCP as well as TCP clamp with a 10KB window for a range of different transfer sizes. Note that the transfer times shown also include the TCP connection establishment and termination overhead. Given the high latency of the link, this overhead can be quite large for short transfers.

TCP clamp does not perform quite as well as expected due to the Linux 2.4.16 receiver offering an initial receive window of just 5392 bytes. Normally, this small window size is not a problem as the receiver rapidly opens the receive window as data starts to flow, and thus does not impede slow-start. However, when using a clamped `cwnd` the receive window is a limiting factor. Despite this, our results demonstrate that TCP `cwnd` clamp provides clear performance benefits for small downloads on GPRS links. We have not found any other TCP stacks that exhibit this initial receive window limitation.

This benefit will be maintained and even enhanced when using HTTP/1.1 persistent TCP connections. When using persistent connections it is normally the case that the server has to let the TCP connection go idle between object transfers since pipelining is rarely supported. Normally this results in the congestion window being set back to its initial value. TCP clamp avoids this, and the benefit is more pronounced due to the lack of connection establishment and termination phases.

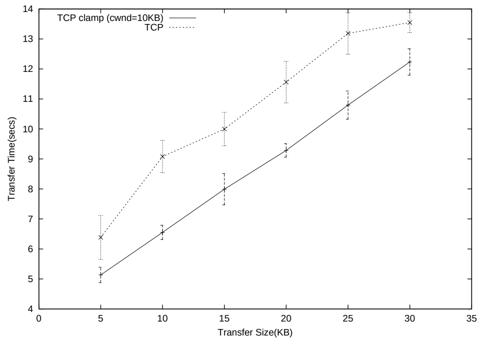

Fig. 11. Results of the `ttcp` download transfers conducted over GPRS network. Plot shows the transfer times for different transfer sizes for TCP clamp (cwnd=10KB) and standard TCP. The error bars correspond to the standard deviation. Each transfer test was repeated 25 times for a given size.

3. Recovery with TCP clamp:- As shown from the TCP sender trace in figure 12(d), an RTO occurs during the file transfer, resulting in packet re-transmission. In this case, there are no data packets in the router buffers, so TCP recovers quickly from the link loss after its first retransmission, and then proceeds with normal data transmission. Without the proxy, there are likely to be a large number of TCP packets queued up over the link before the timeout. This is particularly unfortunate if either host does not support SACKs, in which case the backlogged packets will be needlessly retransmitted. Worse, acks of the retransmitted segments could trigger further retransmissions due to dupacks, leading to a cycle of spurious retransmissions. By limiting the outstanding data over the link the recovery phase is enhanced and occurrence of such spurious retransmission cycles are avoided.

B. Application Level Enhancements (AL-E)

Extending it traditional role as a caching proxy, we have modified squid (v2.4) to provide an additional application level enhancement. The modification allows squid to accept HTTP/1.1 pipelined connections from pipelined capable web clients (browsers).

C. Quantifying the Benefits of our scheme

To evaluate the performance benefits of our scheme we perform experimental downloads over GPRS using both static and dynamic web content. For static web content we make use of the a test web-site that we describe in a later section. For the dynamic web-pages, we created a mock-up of a popular news web-site, CNN `www.cnn.com`, on a locally provisioned web server. This was done to avoid the effects of fast changing web-content and to eliminate

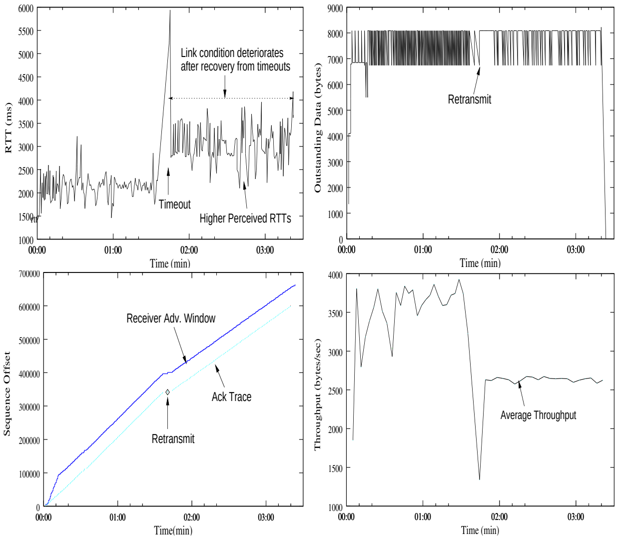

Fig. 12. Early recovery from TCP timeout during 600KB file transfer with TCP clamp. Plots showing (top-left and clockwise) (a) RTT plot of TCP timeout (b) Outstanding (inflight) data (c) receiver perceived throughput and (d) sender trace. Reducing queued data can help TCP to recover quickly. This approach also ameliorates the problem of spurious retransmissions due to sudden delays.

measurement 'noise' that could be introduced by changes in Internet performance. We refer to our dynamic web-site as local CNN, or in short LCNN, to reflect the locally provisioned content from the CNN web-site.

C.1 Test Web Site offering Static Web Content

For the static test web-site, we adhere to a approach very similar to the one used in [11], *i.e.* we compose a number of objects from other web-sites. Simple web sites typically have a base HTML documents along with many embedded or *inlined* objects (gifs, CSS, scripts etc). We have observed that popular sites often have a base page of about 40-50KB with references to often over 50 embedded objects. To offer better control over content presentation, these web sites also often make use of cascading style sheets (CSS) and scripts.

Resource Type	Size Range	# of files
index.html	40K	1
jpegs/gifs	200B-2KB	20
gifs	2KB-5KB	20
gifs	5KB-10KB	4
gifs	>10KB	1

Table 1. Composition of our reference Test static content Web-Site

For the static test, we have synthesized pages based on object type and size distributions observed in HTTP log traces (for example, see table 1).

C.2 Browser Selection

To compare download performance of our scheme, we chose mozilla [44]. The latest release from mozilla 5.2 (developer build version) also supports pipelining. However, after analysing browser traces we found few instances where the browser actually pipelined requests on persistent connections, even when doing so was clearly possible. We expect future Mozilla versions will have a request scheduler better optimised for pipelining.

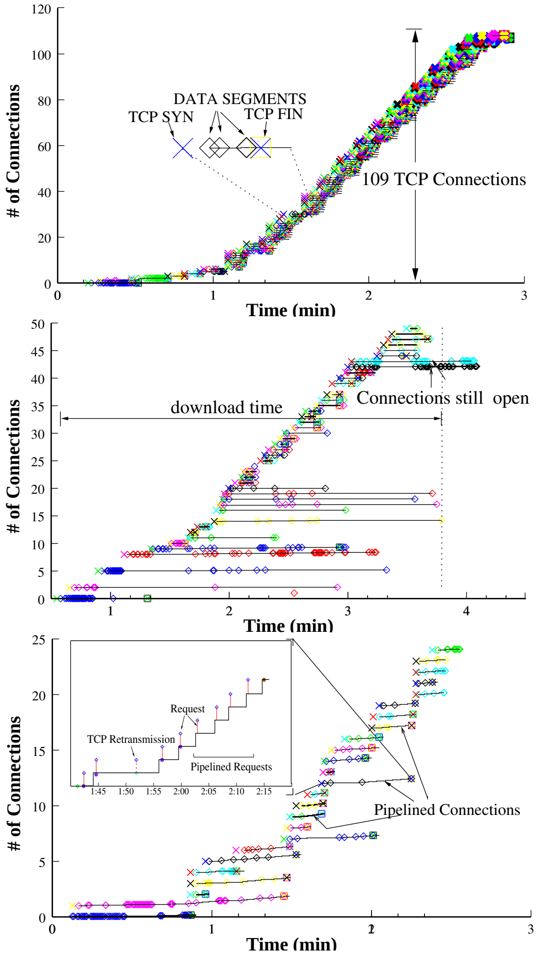

Fig. 13. Mozilla Connnection Timelines for our LCNN web-site, using (top-bottom) (a) non-persistent connection mode, (b) persistent connection mode and (c) pipelined connection mode.

In persistent connection mode, which is the default setting, Mozilla makes use of a maximum of 6 simultaneous connections to an intermediate proxy. In non-persistent mode, it can use a maximum of 8 parallel connections via a proxy.

C.3 Impediments in Measuring Browser Download Times

Measuring download times with a browser is not as easy as it at first appears: Many browsers keep connections open even after the complete page is downloaded. While this makes little difference to a user surfing for some information, it impedes accurate measurements of web site download times. To overcome this problem, we make use of *browser timelines*. Browser timelines are plots that indicate connection timelines made by a browser *i.e.* the number of connections, connection start and end points (if a end point exists), number of requests made, and data received on each request-reponse exchange. We have developed a tool (*timeline* [17]) that uses `tcpdump` and `tcptrace` infor-

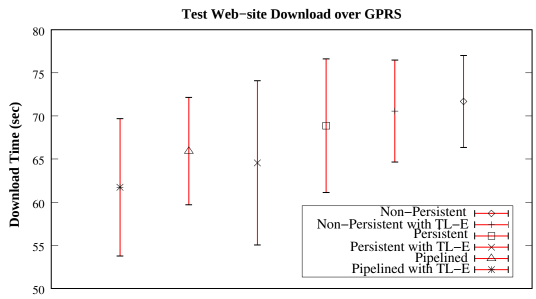

Fig. 14. Test Web Page download time over GPRS using Mozilla

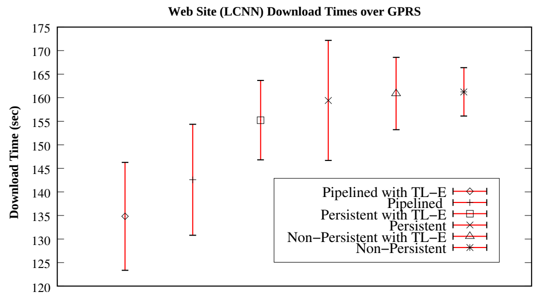

Fig. 16. LCNN download time over GPRS using Mozilla.

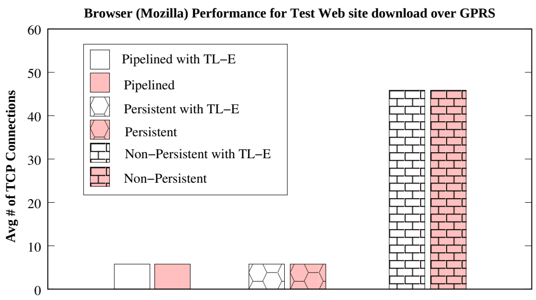

Fig. 15. Avg. Connections to download Test web site using Mozilla

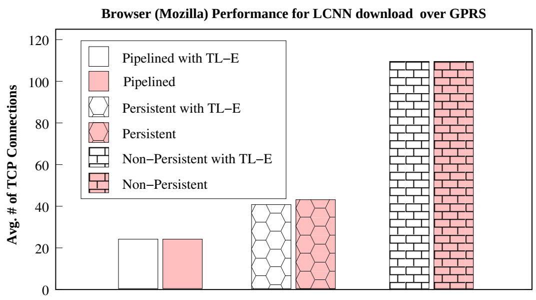

Fig. 17. Avg. Connections to download LCNN using Mozilla

mation to plot browser connection timelines.

Figure 13 shows the sample browser timelines for LCNN web-site using mozilla. The connection timelines are shown for download of LCNN web-site using browser's non-persistent connection mode (figure 13(a)), persistent connection mode (figure 13(b)) and pipelined connection mode (figure 13(c)). The figure (b)-(c) show steps in connection timelines indicating requests sent over an existing connection to be reused in persistent and pipelined connection mode repectively for multiple request-response exchanges. As shown in figure 13(b), some browser connections are kept open even after the download completed.

In some cases, this is as a result of the browser hoping it can re-use the connections to fetch the next page. Such connections are usually timed out by the server after e.g. 60 seconds. In other cases, the connection is kept open as a result of scripts periodically refreshing information (e.g. a stock ticker). For all such cases, we simply measured the download time with respect to connections that finished successfully prior to those that were kept open. This is shown in figure 13(b) with a dotted vertical line. Also, shown in figure 13(c) is the close-up of a connection in which requests were pipelined. The connection starts with a persistent mode and then switches to pipeline requests over the given connection. It is evident here that mozilla pipelined requests only over selected connections from the whole connection pool, giving a relatively poor pipelining efficiency. The best-case browser pipelined-connection (the one shown zoomed in 13(c)) had a ϕ of less than 3, with overall pipelining efficiency (η) of the browser in this case less than 9% (only marginally better than the persistent case).

C.4 Measurements over GPRS Test-Bed

We performed experimental downloads with mozilla using its three different modes: *non-persistent* mode, *persistent* connection mode and *pipelined* connection mode. All experiments were conducted seperately for the static test web site as well as for our locally available LCNN web site offering dynamic content.

We averaged download times from about 20 sucessful runs and plot mean value of download time and corresponding standard deviation. Figure 14 shows that mean download time using mozilla for our test web-site. With non-persistent connections, we found only a small advantage in using the proposed transport level enhancement (TL-E). It seems that for non-persistent connections, overall gains made by eliminating slow-start is negated by the overhead due to large number of connections. Nevertheless, there is still some modest improvement in download times when using TL-E with persistent connections. The average improvement in download times when switching from non-persistent to persistent mode and using TCP *cwnd clamping* was more than 10%. As far as the number of connections are concerned (see figure 15) the reduction was more than 80%. For pipelined connections, as evident from figure 14, we find that using TL-E can reduce mean download time by about 20%.

However, we found that browser traces indicate only few intances of connections over which requests were pipelined. We believe that since the pipelining efficiency of the browser was low (as indicated earlier from browser timelines in figure 13(c)), only meagre benefit could currently be extracted using pipelined connections.

In contrast, observations for the LCNN web site are somewhat different. The use of TL-E with non-persistent connections show very little performance gains. The same is true for persistent connections that show only slight performance improvement in mean download times when compared to non-persistent connections. We believe that since the number of connections made is relatively large for the browser even with persistent mode (an average of more than 40 connections, see figure 17) the gain offered by eliminating slow-start is again negated by a high connection overhead.

However, using the browser's pipelined mode with LCNN lowers the number of connections utilized and also indicates (using browser traces) a somewhat better pipelining efficiency when compared to the web site having static web content. Figure 16 shows that an improvement with pipelined mode was more evident when using the transport level enhancement (TL-E). We found an average reduction of more than 15% in mean download times of the web browser when using the pipelined mode with TL-E. The average number of connections is reduced from 109 in non-persistent mode to an average of just 24 (see figure 17).

An average 15-20% improvement in mean download times for both static and dynamic content over GPRS with pipelined connections is encouraging, taking into account the low pipelining efficiency in mozilla. We believe that greater benefit can be achieved if browsers pipeline their requests more aggressively. Furthermore, these measurements were taken under good radio conditions. The TCP-layer enhancements result in faster recovery from packet loss, which will provide additional improvement in typical usage.

Further, we find that a reduction in the number of connections is advantageous not only due to low-bandwidth nature of the GPRS links but also because it mitigates the overall control and associated transactional (3-way TCP handshake) overhead with each additional connection. Thus we infer that aggressive pipelining of requests over browser connections reduces overall connection (control and transactional) overhead, and combined with the transport level enhancement (TL-E) can result in significant improvement in web download times.

VII. Related Work

A. Measurements of other Wide-area Wireless networks

Measurements taken from a number of other wide-area wireless networks, including GPRS, are available [8], [14], [15], [6], [38].

For GPRS-related measurements: A. Gurtov *et al.* [38] show RTTs for an unloaded GPRS network in Finland to be in the range of 500ms-1100ms with a typical value of around 700ms. They claim an improvement of about 200ms from measurements last conducted in [39]. They validate presence of high amount of GPRS downlink buffering – of about 50kbytes. P. Stuckmann *et al.* in [6] conduct similar measurements in Libertel-Vodafone GSM/GPRS network in Maastricht, The Netherlands. The tests, though not extensive, indicate a very high initial RTTs (> 1s) by using pings that also substantiate our claim of high round-trips in GPRS.

For other networks, P. Sinha *et al.* present measurements for wide-area CDPD network [33]. They show that CDPD links have high RTTs (800ms-4sec.) and low measured raw throughputs (of about 19kb/s). Each CDPD channel can be shared by 1-30 users, while effective throughput of a channel would not exceed 12kb/s. Elan and Hari Balakrishnan evaluated performance of Metricom Richochet wide-area wireless network [24], where minimum RTTs of the order of 280-300ms were found under lightly loaded conditions. This could go as high as 300ms to 5s during loaded condition (e.g. during bulk transfer). TCP throughput was measured between 15kb/s and 30kb/s for various packet sizes while raw (UDP) throughput was measured between 50-58 kb/s (with no reverse channel contention). TCP performance, in general, degrades due to reverse channel contention and large RTT variations (between 250 to 5000 ms).

T. Alanko *et al.* in [14] present link characterization results for GSM networks. Their findings show that GSM links have high RTTs (>1s) with transfer rates around 700-900 bytes/s (without compression). GSM links were found to be relatively stable and reliable but sensitive to external noise and interference that could lead to long delays. When affected by such environmental factors, file transfer peformance was hindered. A drawback with their measurement was that it was made in a "normal office environment", hence results are susceptible to large deviations. Furthermore, measurements were made with a single active connection over the GSM link, and performance with several active connections was not evaluated.

G. Xylomenos *et al.* [8] measured RTTs (using `ping` tests) for a number of GSM networks located in Berlin, Oulu and Helsinki and found RTTs to be higher than 600ms and standard deviation more than 20ms. Large file transfer experiments revealed higher RTTs (up to 12s), which is of similar order to that of GPRS. Typical link level (RLP-level) disruptions in GSM (link level resets or some serious protocol violations) were shown to last for couple of seconds, which disrupts TCP transfers for over 6-12 seconds. To reduce link level resets, it claims that maximum number of re-transmission rounds can be increased. Another interesting revelation in [8] was the evidence of out-of-order packets released in GSM (which means GSM RLP is not fully reliable), contrary to our experience with GPRS where we have recorded no such occurance of packet-level reordering. In [15], R. Lugwig *et al.* show that GSM RLP usually recovers from losses before TCP timers can expire, but suffer from high variability in RTT values. This concurs with our observation of GPRS networks. Evidence of TCP *spurious timeouts* and *spurious fast re-transmits* were also shown to occur over GSM.

B. TCP Performance over Wireless

In this section, we quickly review TCP enhancement schemes over wireless. A plethora of solutions exists for elevating TCP performance over wireless links; however, none

Proposals	SNOOP[22]	M-TCP[32] & I-TCP[31]	Freeze-TCP[34]	FDA[36]	WTCP[33]	W-TCP[26]
TCP change in end-systems required	no	no	yes	no	yes	no
Avoids Excess Queuing at Proxy/Base Station	no	no	no	yes	yes	no
Faster Start-up for short flows	no	no	no	no	yes	no
Maintains TCP Fairness for all types TCP flows	no	no	no	no	no	no
Handle variable bit error environment	yes	yes	yes	no	yes	yes
Handle small link "stalls"	no	yes	no	no	yes	yes
Quick Recovery from Long Disconnections	no	yes	yes	no	no	yes
Maintains End-to-End Protocol Semantics	yes	no	yes	yes	yes	yes

Table 2. TCP Enhancements over Wireless – A Comparision

of the schemes have attempted to improve start-up performance for TCP flows. A careful examination of the existing schemes suggests four broadly different approaches to improve TCP performance: link-layer based schemes (both TCP aware and unaware) (e.g. [22], [35]), split connection based approaches (e.g. I-TCP [31], W-TCP [26]) and early warning based approaches (e.g. FreezeTCP [34]) and finally those necessitating end system changes (e.g. WTCP [33], Freeze-TCP [34]).

Snoop [22] 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 they are present in a 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. The Snoop protocol scheme was originally designed for wireless LANs rather than 'long-thin' wide-area wireless links. As such, it does not address the problems of excess queueing at base stations or proxies. Jian-Hao *et al.* developed FDA [36], which uses a Snoop-like strategy, but uses a novel flow-control scheme which goes some way to prevent excess queueing. Delayed DupAcks [35] is a TCP unaware link-layer scheme: during radio losses, it suppresses DupAck's for some interval d, which unfortunately is difficult to determine.

The second broad approach is to split the TCP connection into two sections. This allows wireless losses to be completely shielded from the wired ones. I-TCP [31] uses TCP over the wireless link albeit with some modifications. Since TCP is not tuned to the wireless link, it often leads to timeouts eventually causing stalls on the wired side. Due to the timeouts, valuable transmission time and bandwidth is also wasted. I-TCP also lacks an appropriate flow control scheme. M-TCP [32] is similar to I-TCP except it better preserves end-to-end semantics. M-TCP uses a simple zero window ACK scheme to throttle transmission of data from the wired sender. This leads to stop-start-stop bursty traffic on the wired connection, and the lack of buffering in the proxy can lead to link under-utilization for want of packets to send.

Ratnam and Matta propose W-TCP [26], which acknowledges a packet to the sender only after receiving an acknowledgement from the mobile host. W-TCP changes the timestamp field in the packet to account for the time spent idling at the base station. On the other hand, WTCP [33] is an end-to-end scheme which primarily uses inter-packet separation as the metric for rate control at the receiver. Congestion related loss detection is also provided as a backup mechanism. The drawback in WTCP is that it entails end-hosts TCP changes.

The third broad approach identified covers schemes that use various kinds of early warning signals. Freeze TCP uses Zero Window Probes (ZWP) like M-TCP, but is proactive since the mobile host detects signal degradation and sends a Zero Window Warning Probe. The warning period *i.e.* the time before which actual degradation occurs should be sufficient for the ZWP to reach the sender so that it can freeze its window. The warning period is estimated on the basis of RTT values. One pitfall is the reliability of this calculation and Freeze-TCP's inability to deal with sudden random losses. Furthermore, Freeze-TCP requires end-system changes.

C. Wireless Web Performance

A number of significant research studies have looked into the web performance, in general [10], [11], [13], and more specifically into wireless web performance [29], [23], [30].

J. C. Mogul *et al.* [10] discuss HTTP throughput and latency problems and show that each document and inline image requires a minimum of ($2 \times$ RTT) for transfer. They also propose the use of persistent connections and pipelining to improve overall web performance. H. Nielsen *et al.* in [11] clearly demonstrate benefits of pipelined implementations, while J. C. Mogul *et al.* in [13] show that use of delta encoding and compression between client browser and a proxy can 'remarkably' improve web performance.

M. Liljeberg *et al.* developed Mowgli system [30] for use with GSM, which uses its own proprietary protocol called Mowgli HTTP (MHTTP), to improve upon the limitations of TCP and HTTP over GSM. The protocol optimizes use of bandwidth using binary encoding in MHTTP, compresses text and images and also uses selective filtering for added optimization. T. B. Fleming *et al.* in [29] use a similar scheme but also include prefetching schemes in their wireless world wide web proxy server and protocol using their new Multiple Hypertext Stream Protocol (MHSP). However, both the schemes necessitate a client side software update. Explicit Loss Notification (ELN) [23] improves web performance using a scheme by which senders are informed about the correct reason for a loss or reasons unrelated to network congestion (e.g., due to wireless bit errors). Thus it decouples sender retransmissions from congestion control.

Commercial products that improve upon the current GPRS performance are also available. A GPRS Accelerator launched by Firsthop [42] 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 [13] or other sophisticated data (and header) compression techniques or by simply filtering/transcoding at the proxy. Nevertheless, either of these approaches would necessitate a client-side software update. This approach is contrary to the one used in our mobile proxy, which clearly obviates the need for any client-side stack or software update. A client-side update that makes use of delta compression over the wireless link could prove highly useful to low bandwidth GPRS users. In the same way, data transcoding proxies can also be advantageous to low-end devices.

It is questionable if pre-fetching schemes over GPRS could be useful. However, a problem germane to predictive prefetching as compared to the deterministic ones is that the former operates on 'guesses' that might waste the limited wireless link bandwidth for needless downloads. This is certainly an expensive proposition to the GPRS end-users, since wireless bandwidth is costly and most pricing schemes are *volume* based.

VIII. Summary and Future Research

In this paper, we presented the results of GPRS link characterization measurements over a real GPRS test bed. These characterization results are based on the measurements conducted over Vodafone UK's national GPRS network, and from measurements conducted over other GPRS networks located in UK and other European countries, using a number of GPRS handsets from different vendors. We have conducted experiments to examine the performance of protocols like TCP and web performance (HTTP) over GPRS, and have suggested and evaluated improvements. We summerize the results of this paper as follows:

- GPRS round-trip times are large and variable. When packets are sent in a burst, only the first packet tends to experience high jitter.
- the throughput available over GPRS is highly variable and can fluctuate rapidly, depending upon the radio conditions.
- brief link outages are a commonplace, even when the mobile host is stationary.
- packet loss over GPRS is relatively rare, due to the robust error recovery employed in the RLC (Radio Link Control) layer. No packet re-ordering instances have been observed.
- TCP over GPRS achieves poor performance for short connections due to the pessimistic nature of the slow-start coupled with high link RTTs. TCP SACKs was found to be highly effective over "long-thin" GPRS links, as it improves link utilization by avoiding unnecessary re-transmissions of packets during loss recovery. The in-order nature of GPRS RLC packet delivery makes it quite TCP friendly.
- Excess queuing at the GPRS CGSN node with long-lived flows or multiple flows can result in a grossly unfair distribution of bandwidth. This can happen while using an aggressive web browser, where a long flow might lead to excess queuing over the link, impeding transfer of 'important' objects over the link.
- An enhanced HTTP proxy located close to the wired-wireless border improves Web performance without necessitating any client-side software updates. A static congestion window control technique in TCP was found to be fruitful, and use of pipelined connections is certainly advantageous to web performance over GPRS.

In the future, we intend to implement and evaluate other optimization schemes such as delta encoding, data transcoding/filtering as well as deterministic prefetching to see what quantitative benefit could be achieved by using such schemes in order to improve web performance over GPRS. These would require modifications to client software.

We are also implementing our transport level enhancements in a generic transparent TCP proxy that will provide benefit to all TCP flows (e.g. IMAP/POP3, FTP etc) rather than just HTTP flows. We are investigating improved error recovery mechanisms and flow control techniques to use in this proxy.

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

Acknowledgments

The authors gratefully acknowledge Vodafone Group R&D, Sun Microsystems Inc. and Benchmark Capital for supporting this work.

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] B. Walke, Mobile Radio Networks, Networking and Protocols (2. Ed.), John Wiley & Sons, Chichester 2001

[3] 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.

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

[5] P. Stuckmann, "Quality of Service Management in GPRS-Based Radio Access Networks", *Telecommunication Systems*, Kluwer Academic Publishers, 19:3,4, pages 515-546, 2002

[6] P. Stuckmann, N. Ehlers, B. Wouters, "GPRS Traffic Performance Measurements", In Proceedings of the IEEE Vehicular Technology Conference (Fall VTC 2002), Vancouver, Canada, Sepember 2002.

[7] J-H. Park, "Wireless Internet Access for Mobile Subscribers Based on the GPRS/UMTS Network", *IEEE Communications Magazine*, April 2002.

[8] George Xylomenos et al., "TCP Performance Issues over Wireless Links", *IEEE Communications Magazine*, April 2001.

[9] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgement Options". *Request for Comments (RFC) – 2018*, April 1996.
[10] V. N. Padmanabhan and J. C. Mogul, "Improving HTTP Latency", *Computer Networks and ISDN Systems*, vol. 28, pp. 25-35, 1995
[11] 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
[12] J. C. Mogul, "Support for out-of-order responses in HTTP", *Internet Draft*, Network Working Group, 6 April 2001.
[13] 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.
[14] T. Alanko, M. Kojo, H. Laamanen, M. Liljeberg, M. Moilanen, and K. Raatikainen, "Measured Performance of Data Transmission Over Cellular Telephone Networks", In *ACM Computer Communications Review*, 24:5, 1995
[15] R. Ludwig et al., "Multi-Layer Tracing of TCP over a Reliable Wireless Link", In *Proceedings of ACM SIGMETRICS*, 1999.
[16] B. Knutsson and L. Peterson, "Transparent TCP Proxy Signalling", *Journal of Communications and Networks (JCN)*, March 2001.
[17] R. Chakravorty, S. Katti, J. Crowcroft and I. Pratt, "Flow Aggregation for Enhanced TCP over wide-area wireless", to appear in *IEEE INFOCOM 2003*.
source: http://www.cl.cam.ac.uk/users/rc277/gprs.html
[18] R. Chakravorty, J. Cartwright and I. Pratt, "Practical Experience With TCP over GPRS", In Proceedings of *IEEE GLOBECOM 2002*, November 17-21, Taipei, Taiwan
source: http://www.cl.cam.ac.uk/users/rc277/gprs.html
[19] 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
source: http://www.cl.cam.ac.uk/users/rc277/gprs.html
[20] D. Dutta and Y. Zhang, "An Active Proxy Based Architecture for TCP in Heterogeneous Variable Bandwidth Networks", In *Proceedings of IEEE GLOBECOM*, November 2001.
[21] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Trans. on Networking*, Vol. 5, No.6, Dec. 1997.
[22] H. Balakrishnan, R. Katz and S. Seshan, "Improving TCP/IP performance over Wireless Networks", In *Proceedings of ACM MOBICOM*, November 1995
[23] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance" In *Proceedings IEEE Globecom Internet Mini-Conference*, Sydney, Australia, November 1998.
[24] E. Amir and H. Balakrishnan, "An Evaluation of the Metricom Ricochet Wireless Network", CS 294-7 Class Project, UC Berkeley. Source: http://www.lariat.org/Berkeley/paper.html
[25] L. Zhang, S. Shenker, D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic", In *Proceedings of ACM SIGCOMM*, 1991.
[26] K. Ratnam and I. Matta, "W-TCP: An Efficient Transmission Control Protocol for Networks with Wireless Links", In *Proceedings of Third IEEE Symposium on Computer and Communications (IEEE ISCC)*, 1998.
[27] Z. Wang and P. Cao, "Persistent Connection Behaviour of Popular Browsers", http://www.cs.wisc.edu/cao/papers/persistent-connection.html
[28] F. Donelson Smith et al., "What TCP/IP Protocol Headers can tell us about the Web", In *Proceedings of ACM SIGMETRICS*, 1999.
[29] 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
[30] 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
[31] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts", In *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 136-143, Vancouver, BC, May 1995.
[32] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks", *Computer Communication Review*, 1997
[33] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bhargavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", In *Proceedings of ACM MOBICOM*, 1999.
[34] 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.
[35] V. Bakshi, P. Krishna, N. H. Vaidya, D. K. Pradhan, "Improving Performance of TCP over Wireless Networks", *Texas A&M University Tech. Report TR-96-014*, May 1996
[36] J. Hu and K. L. Yeung, "FDA: A Novel Base Station Flow Control Scheme for TCP over Heterogeneous Networks", In *Proceedings of IEEE INFOCOM*, 2001
[37] J. Cartwright, GPRS Link Characterization,
Source: http://www.cl.cam.ac.uk/users/rc277/linkchar.html
[38] A. Gurtov, M. Passoja, O. Aalto, M. Raitola "Multi-Layer Protocol Tracing in a GPRS Network", In Proceedings of the IEEE Vehicular Technology Conference (Fall VTC2002), Vancouver, Canada, Sepember 2002.
[39] J. Korhonen, O. Aalto, A. Gurtov, H. Laamanen, "Measured Performance of GSM HSCSD and GPRS", In Proceedings of the IEEE ICC 2001.
[40] An Introduction to the Vodafone GPRS Environment and Supported Services, Issue 1.1/1200, December 2000, Vodafone Ltd., 2000.
[41] tcpdump(http://www.tcpdump.org),
tcptrace(http://www.tcptrace.org),
ttcp+(http://www.cl.cam.ac.uk/Research/SRG/netos/netx/)
[42] Firsthop GPRS Accelerator, http://www.firsthop.com/
[43] The squid proxy cache Homepage, http://www.squid-cache.org
[44] Mozilla web browser, http://www.mozilla.org
[45] Opera web browser, http://www.opera.com

Rajiv Chakravorty is a second year PhD student at the University of Cambridge Computer Laboratory, where he holds the SUN Microsystems student Fellowship and a Hughes Hall CommonWealth Scholarship from his college, Hughes Hall. Prior to joining for his PhD, he was working with Philips Research - ASA Labs at Eindhoven in The Netherlands. He received his MTech from Indian Institute of Technology (IIT), Delhi and a B.E. from Nagpur University in 1999 and 1997 respectively. At IIT Delhi, he was a recipient of DAAD Scholarship award from Germany for which he carried out his research at the Chair of Communication Networks (ComNets), RWTH-Aachen, Germany. His interests broadly include Mobile and Wireless Communications, Ubiquitous Systems and Quality of Service.

Ian Pratt is a Senior Lecturer at the University of Cambridge Computer Laboratory. He holds a PhD in Computer Science and was elected a Fellow of King's College, Cambridge in 1996. As a member of the Lab's Systems Research Group for over 10 years, he has worked on number of influential projects, including the Fairisle ATM LAN, the Desk Area Network workstation, and the Nemesis operating system. His research interests cover a broad range of Systems topics, including computer architecture, operating system design, mobile communications, and networking.