Practical Experience with TCP over GPRS

Rajiv Chakravorty, Joel Cartwright and Ian Pratt
{rajiv.chakravorty,joel.cartwright,ian.pratt}@cl.cam.ac.uk
University of Cambridge Computer Laboratory, Cambridge CB3 0FD, U.K.

Abstract— **We present the results of a series of experiments used to characterise the performance of a GPRS (General Packet Radio Service) wireless data network, hence highlighting issues that software architects should consider when designing applications to run over this soon to be widely-deployed service. In summary, we show that packet Round Trip Time (RTTs) are large (>1000ms) and can be highly variable, packet losses are relatively rare, and that available bandwidth can be quite variable.**

These network characteristics do not interact well with current TCP implementations. We show how it takes many seconds before a new TCP connection can expand its congestion window to make use of the full bandwidth available, leading to very poor performance of protocols like HTTP. Beyond the point of full bandwidth utilisation, TCP continues to expand the window needlessly, resulting in excessive queueing at the GPRS router. This leads to greatly inflated RTTs (10's of seconds) and hence poor interactive response and slow recovery should loss occur. We show how a simple transparent proxy interposed between the fixed and GPRS networks can be used to significantly improve TCP connection performance, particularly for activities like web browsing.

I. Introduction

The past few years have witnessed substantial growth in cellular telephony, which has also lead to an increased demand for wireless data services. Although now an agreed standard, so-called 3rd generation (3G) UMTS (Universal Mobile Telecommunications System) mobile networks are still some time away from wide-scale deployment. In the meantime, the GPRS (General Packet Radio Service) extension to current "2G" GSM networks looks set to provide a widely deployed solution for wireless data access.

Surprisingly, there have been few published studies of the performance of Internet protocols operating over GPRS links. Most work in this direction has focused on simulated networks [1][4]. In contrast, the measurements presented in this paper have been taken on a nationally deployed commercial GPRS network.

In particular, we examine the behaviour of TCP, the dominant protocol in the wired Internet, and one that is likely to be similarly important for mobile users too. Although TCP in wireless LAN environments has been widely studied [9], the characteristics of GPRS lead to a different set of pressing performance issues. We go on to show how interposition of a transparent proxy can be used to mitigate some of these effects and yield better TCP throughput over GPRS links without modification of the hosts.

II. GPRS Overview

GPRS is a bearer service for GSM - a wireless extension to packet data networks. It enables an "always-on" service where resources (time-slots) are consumed only when data packets are actually transmitted. 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.

A reliable RLC (radio link control) mode ensures that packets are delivered in order, while the ARQ (automatic repeat request) in RLC combined with FEC (forward error correction) helps to recover from packets received in error. GPRS copes with a wide range of radio conditions by making use of 4 different coding schemes (CS-1 to CS-4) [1][4] with varying levels of FEC. Most currently deployed GPRS networks support only CS-1 and CS-2 [6] – the other two are not used as error rates would be typically too high to be useful. The CS-2 scheme employs a coding rate of approximately 2:3, and obtains a transmission rate as high as 13.4 kbit/s per GSM time slot [4]. The effective GPRS data rate will be 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 between mobile terminals with data to send or receive.

Mobile terminals are classified according to the number of time slots they are capable of operating on simultaneously. For example, most current devices are classified as '3+1' meaning that they can simultaneously listen to 3 downlink channels (from base station to mobile), but only transmit on 1 uplink channel to the base station. Assuming CS-2 coding is in use, this corresponds to a maximum downlink bandwidth of 40.2 Kbit/s and uplink bandwidth of 13.4 Kbit/s.

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 levels. In practise, most operators only support a single 'best effort' service.

Further information about GRPS operation can be found in [1].

III. Measurements for Link Characterization

A. Test Bed Setup

Our experimental test bed for characterizing GPRS links is shown in figure 1. The measurements presented in this paper were all performed over Vodafone UK's GPRS network, though we have also observed comparable results using BT Cellnet's network. In the test set-up, 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. We have also used 3+1 channel phones from other manufacturers and obtained very similar results.

Complying to the usual GPRS architecture, the base stations (BSs) are linked to the SGSN (Serving GPRS Support Node) which is then connected to a GGSN (Gateway GPRS Support node). In the current Vodafone configuration, both SGSN and GGSN node is co-located in a CGSN (Combined GPRS Support Node). A well provisioned virtual private network (VPN) connects the Lab network to that of the Vodafone's backbone via an IPSec tunnel over the public Internet. The RADIUS server is used to authenticate mobile terminals and assign IP addresses.

We have configured routeing within the Lab such that all traffic going to and from mobile clients must pass through a Linux-based software router. This enables us to perform traffic monitoring as well as providing a location to run the transparent mobile proxy we describe later.

Link characterization measurements were performed during the night to reduce the possibility of contention for GSM time slots with other users. However, repeating the experiments at various times during the working day revealed no sign 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.

B. Preliminary Results from GPRS link characterization

The tests were performed using a version of the `ttcp` program modified (`ttcp+` [11]) to enable traffic streams to be generated at specified rates and with particular burst characteristics. We also insert time

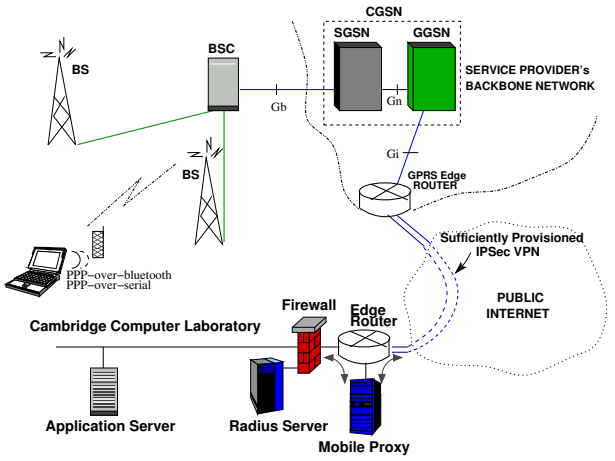

Fig. 1. Test Bed Set-Up for Performance Measurements

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.

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. A separate technical report containing a more detailed discussion of GPRS link characteristics is also available [5].

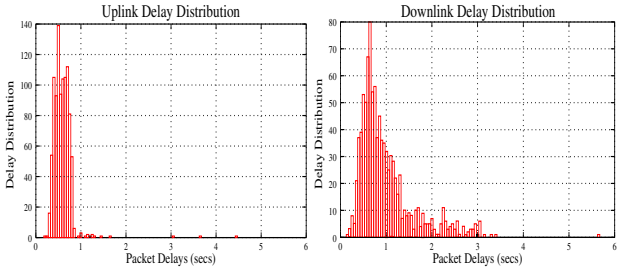

Fig. 2. Plots showing delay distributions (non-normalized) for 64 bytes packet case (a) uplink delay distribution (b) downlink delay distribution. Measurements involved transfer of 1000 packets, rate limited to a 4 seconds delay between two successive transfers. Various other scenarios for different packet sizes are also available in [5].

Figure 2 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 very poor for any kind of interactive application.

If the experiment is 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 initial jitter will in part be due to the 'paging channel' required to inform a mobile terminal that data is to be sent to it.

The latency figures reported here (measured on two different GPRS networks) are rather higher than predicted in previous simulation studies. We are currently in discussions with the network operator to try and determine the source of the discrepancy.

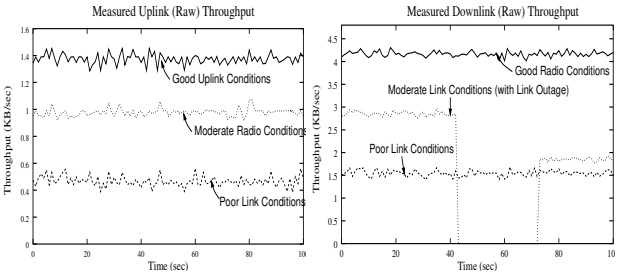

Fig. 3. Raw throughput measurements using Mototola (3+1) GPRS phone for the case of (a) uplink and (b) downlink.

Raw UDP bandwidth was measured using `ttcp` 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 (5 secs) enable variations in raw link bandwidth to be observed (packet losses in this experiment are ignored as they are most likely due to packet discard at the CGSN).

Figure 3 shows raw UDP bandwidth traces for the uplink and downlink directions taken under a number of different radio conditions. Note how the available bandwidth often varies with time as radio conditions change (we believe there was no contention for time slots).

In particular, note how the 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 outages are not uncommon, particularly when the mobile terminal is on the move in a car or train. Link outages typically last 5-40s. However, due to the RLC's ARQ protocol packets are rarely lost, just grossly delayed.

Packet loss does occur over GPRS links in both the downlink and uplink directions, but the incidence is relatively rare, and hard to quantify. When loss does occur, it is quite likely to occur in bursts of consecutive packets. The RLC implementation should prevent packets from being re-ordered, and indeed, no re-ordering events have ever been observed.

IV. TCP Performance over GPRS Links

As well as the link characterization measurements, we also performed separate tests to gain a better insight into the TCP performance, specifically over the downlink channels. We targeted the downlink channel because of its importance in Web applications.

In this experiment, file transfer tests were performed during different radio conditions, and traces of the transfer collected using `tcpdump` [11] at both ends – the sender host in the Lab and the mobile receiver laptop connected via a GPRS phone. Both hosts used Linux version 2.4.16, which employs a modern TCP implementation supporting Selective ACKnowledgements (SACKs) [2].

The traces were analysed using `tcptrace` [11]. To understand steady-state link behaviour, we selected a reasonably large file transfer size of 600KB. Figure 4(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 4(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.

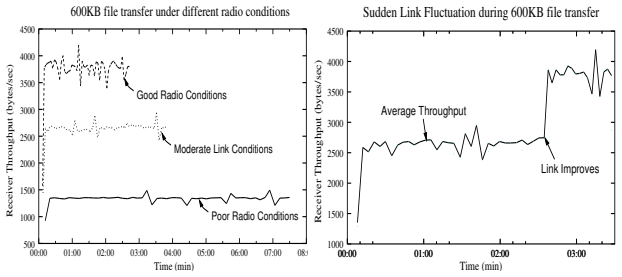

Fig. 4. Plot (a) shows the 600KB transfer progressing under three different radio conditions. Plot (b) shows how bandwidth can change during the course of a transfer

A. The effect of high RTTs on the slow start phase

In figure 5(a) we observe the classic outline of a TCP sender going through slow start (SS) and entering the congestion avoidance phase. What is unusual here is that it is not loss that causes congestion avoidance to be entered, but receiver window limitations. A further point to note about this particular trace 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 compression 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.

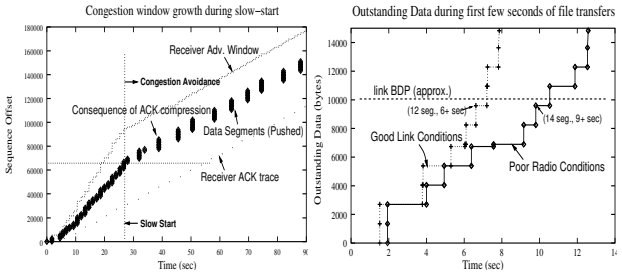

Fig. 5. Plot (a) shows the characteristic exponential congestion window growth due to slow-start (SS). The connection leaves SS due to receive window limitations rather than packet loss. Plot (b) shows that SS takes 6+ seconds before the congestion window is expanded sufficiently to enable the connection to utilise the full link bandwidth.

A close up of the first few seconds of the connection is presented in figure 5(b), alongside another connection under slightly worse radio conditions. Growth of the sender's congestion window in response to ACKs can be observed. Marked on the plot is an estimate of the bandwidth delay product (BDP) of the GPRS link, approximately 10KB. Note this 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.

We see that in the case of good radio conditions this takes approximately 6 seconds from the initial TCP SYN message (longer under worse conditions). Hence, for transfers shorter than about 16KB TCP fails to exploit even the meagre bandwidth that GPRS makes available to it. Since many HTTP objects are around this size the effect on web browsing performance can be dire. Section VI explores this further.

It is worth noting that although Linux uses an initial congestion window value of 2 many other TCP implementations use a value of 1, which would further delay discovery of the correct congestion window.

B. Excess queueing

TCP uses loss as an indication of congestion, and hence a signal that it should halve its congestion window. However, in the case of a GPRS link the connection's bandwidth bottleneck is almost always due to the radio link rather than to multiplexing at a router in the wired Internet. Hence, in the down link direction packets tend to accumulate at GPRS CGSNs. The buffering resources within the CGSN are substantial: Using UDP burst tests we have observed over 120KB of buffering. Given a long enough connection, and assuming no packet losses occur due to radio errors (which are normally fixed-up by the RLC), TCP's congestion control algorithm would fill the entire router buffer before incurring packet loss and reducing its window.

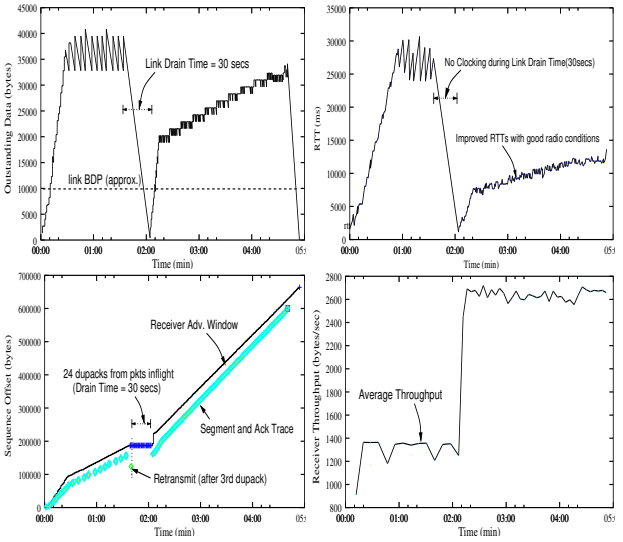

Fig. 6. Case of generation of dupacks (sacks in this case) during 600KB file transfer. Plots showing effect of sacks on (clockwise from top-left) (a) outstanding TCP data (b) sender perceived RTTs (c) receiver perceived throughput (d) sender packet trace. The time to drain data from the link is high, in this case, about 30 secs. The dotted line in (a) gives the estimated value of link BDP.

Fortunately, the window is not allowed to get 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 queueing delay. The left hand sections of figure 6(a) and (b) show a TCP connection is just such a state: There is 40KB of outstanding data leading to a measured RTT of around 30 seconds! Excess queueing leads to a number of problems:

- **RTT Inflation**:- Higher queueing delays can severely degrade TCP performance [3]. A second TCP connection established over the same link is likely to have its initial connection request time-out [8]. Interactive applications become neigh impossible in the presence of a simultaneous bulk transfer.
- **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 [8].
- **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. This is shown in figure 6(a)-(d).
- **Complexities in Network Buffer Provisioning**:- Multiple users saturating a bottleneck link with TCP data may lead to increased complexities in buffer space provisioning.

B.1 Implications of Packet Loss

Results from our link characterization measurements indicate that losses are relatively rare. However, some losses do occur, and can cause TCP problems. Figure 6 shows one such case in which a loss occurred.

The main point to note is the large amount of time (30 seconds) it takes TCP to recover from the loss, on account of the excess quantity of outstanding data. Fortunately, the connection is using SACKs and thus the packets transferred during the recovery period are not discarded, and the effect on throughput is minimal. This emphasises the importance of SACKs in the GPRS environment – it should be implemented even on simple devices like PDAs where it is often currently omitted.

The occasional link outages cause particular problems for TCP. Outages can freeze data tranfer over the link during the outage interval, which can lead to RTO triggered retransmissions that later turn out to be spurious when the data is finally released over the link.

V. Improving TCP performance over GPRS

From our observation of TCP over GPRS, we conclude that the high underlying RTT on GPRS links results in suboptimal perfomance, particularly for short-lived sessions. Furthermore, we have also seen that TCP causes excess queueing, leading to variety of performance problems.

We present a simple technique that improves TCP performance over GPRS without need to modify either the sending or receiving hosts. This is achieved through interposition of transparent 'mobile TCP proxy' running on a Linux router using netfilter [10] to divert the packets to a user-space daemon.

Ideally, the mobile proxy would be co-located with the GPRS GGSN node, but in our implementation we locate it at our end of the IPSec tunnel, where it gets to examine (and can modify) packets going in both directions over the link.

The proxy transparently splits TCP connections [7] into two legs: the 'wired' section and the 'wireless' section. Over the wireless section, the proxy uses a modified TCP sender that uses a fixed size congestion window, the size picked to be the current estimate of the BDP of the link. Thus, slow start is eliminated, and further unnecessary growth of the congestion window is avoided. As a further optimisation we re-write the receiver window advertised in ACKs heading back to the sender to control the amount of data it causes to be queued at the proxy. We call our technique TCP `cwnd` clamping.

Attempting to apply our scheme to the Internet as a whole would certainly be disastrous; slow start and congestion avoidance normally serve essential roles. However, in the GPRS case congestion avoidance is largely redundant. It is possible for the proxy to maintain state about all of the TCP connections heading to a particular mobile terminal and share the BDPs worth of buffering out amongst the connections appropriately. The underlying GPRS network is ensuring that bandwidth is shared fairly amongst users (or according to some other QoS policy), 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 rapidly adjust its fixed size congestion window, but in practise this is currently unnecssary.

The following sections describe our approach in more detail.

A. TCP `cwnd` Clamping

In `cwnd` clamping, the mobile proxy avoids the slow start phase and starts with a congestion window that enables full use of link bandwidth. It uses a clamped value (C_{clamp}) of `cwnd` and maintains it for the full duration of the connection. Once the mobile proxy is succesful in sending C_{clamp} amount of data it goes into a self-clocking state in which it clocks out one segment each time its receives an ACK from the receiver. This approach maintains the amount of outstanding data to an optimistically estimated value of the link BDP, neither significantly overrunning the link, or under utilizing it.

The `cwnd` remains clamped even during times or 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 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 (e.g. 8+1), the proxy may need to use traffic shaping to smooth the initial burst of packets to a conservative estimate of the link bandwidth.

In absence of any queuing or packet loss, the window size necessary to keep a link busy without any idle times should correspond to the BDP of the link. A reasonable value for the clamp window value i.e. C_{clamp} can be made from the maximum values of D_{link} and B_{link} i.e. D_{max} and B_{max}. These values can be obtained through appropriate link capacity measurement techniques. In such a case, the clamp value (C_{clamp}) for the congestion window will be given by $C_{clamp} = D_{max} \times B_{max}$. This value should avoid the link going idle and hence under-utilization.

If the estimate of the BDP is good, packets arriving at CGSN router will experience minimal queueing. As radio conditions vary the amount of queueing may increase, but will be bounded, and is likely to be significantly less than the excesses of normal TCP.

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

VI. Validating TCP `cwnd` Clamp Strategy

In this section, we evaluate the efficacy of TCP `cwnd` clamping. Specifically, we show 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**:- Slow-start is avoided, which improves start-up performance and transfer times (especially for short web transfers).
- **Quick Recovery from Losses**:- It 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 left-over) TCP data due to abnormal disconnections.

A. 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 7(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 which under utilization has occurred due to ACK compression; for clarity the line is omitted.

Higher values of `cwnd` leads to higher values of perceived 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.

B. 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) down-

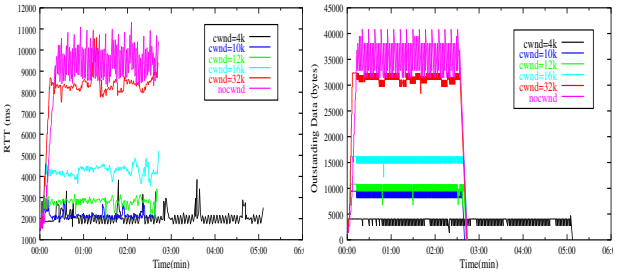

Fig. 7. Figures showing (a) sender perceived RTTs and (b) Outstanding (inflight) TCP data during 600KB file transfer. Effects of queuing delays can be effectively reduced by clamping the congestion window. A good selection of C_{clamp} ensures that link is never underutilized.

loads, primarily to reflect web sessions behaviour. Each transfer for a given size was repeated 25 times and traces recorded using `tcpdump`.

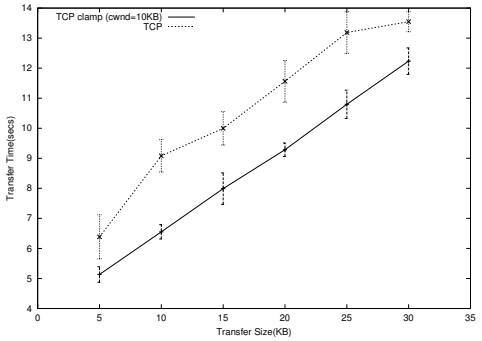

Fig. 8. Results of the `ttcp` download transfers conducted over GPRS network. Plot shows the transfer times for different transfer sizes for TCP clamp (cwnd=10KB) and standard TCP. The error bars correspond to the standard deviation. Each transfer test was repeated 25 times for a given size.

Figure 8 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 expected due to the Linux 2.4.16 receiver offering an initial receive window of just 5392 bytes. The receiver rapidly opens the receive window as data starts to flow, and the results demonstrate that despite the initial window limitation TCP `cwnd` clamp provides clear performance benefits for small downloads on GPRS links.

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.

C. Recovery with TCP clamp

As shown from the TCP sender trace in figure 9(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 would further 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.

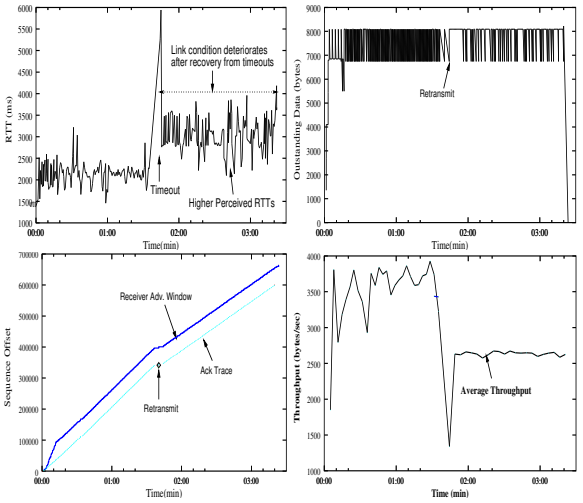

Fig. 9. Early recovery from TCP timeout during 600KB file transfer with TCP clamp. Plots showing (top-left and clockwise) (a) RTT plot of TCP timeout (b) Outstanding (inflight) data (c) receiver perceived throughput and (d) sender trace. Reducing queued data can help TCP to recover quickly.

VII. Conclusions

In this paper, we investigated the performance of TCP operating over GPRS links. In addition to presenting link characterization results, we conducted several file transfer tests to gain useful insight into the typical TCP performance. We quantified the effect of GPRS link under utilization during slow start, and observed the deleterious effects of queueing caused by excessive congestion windows.

We described how a transparent proxy may be inserted into the network close to the wired/wireless boundary to improve TCP performance without modification of the end hosts. We validated our approach through file download trials over GPRS links under different radio conditions and conclude that the proxy boosts TCP performance by reducing queueing delays and improving overall throughput.

Acknowledgments

The authors would like to thank 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] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, “TCP Selective Acknowledgement Options”. RFC 2018, April 1996.
[3] D. Dutta and Y. Zhang, “An Active Proxy Based Architecture for TCP in Heterogeneous Variable Bandwidth Networks”, IEEE GLOBECOM, November 2001.
[4] M. Meyer, “TCP Performance over GPRS”, In Proc. of IEEE WCNC, pages 1248-1252, 1999
[5] “GPRS Link Characterization”, http://www.cl.cam.ac.uk/∼rc277/linkchar.html
[6] “An Introduction to the Vodafone GPRS Environment and Supported Services”, Issue 1.1/1200, December 2000, Vodafone Ltd., 2000.
[7] O. Spatscheck et al., “Optimizing TCP Forwarder Performance”, IEEE/ACM Transactions on Networking, Vol. 8, No. 2., April 2000
[8] R. Ludwig et al, “Multi-Layer Tracing of TCP over a Reliable Wireless Link”, In Proceedings of ACM SIGMETRICS 1999.
[9] H. Balakrishnan et al., “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Trans. on Networking, Vol. 5, No.6, Dec 1997.
[10] The Linux NetFilter Homepage, http://www.netfilter.org
[11] tcpdump(http://www.tcpdump.org), tcptrace(http://www.tcptrace.org), ttcp+(http://www.cl.cam.ac.uk/Research/SRG/netos/netx/)