Path Tail Emulation: An Approach to Enable End–to–End Congestion Control for Split Connections and Performance Enhancing Proxies

Detlef Bosau
detlef.bosau@web.de

Abstract:

Wide area mobile networks facilitate TCP/IP with radio link protocols (RLP) in order to achieve acceptable throughput. Varying channel properties, roaming etc. which interfere with the TCP retransmission and congestion control mechanisms can be alleviated by split connection techniques and performance enhancing proxies (PEP).

However, current split connection techniques and PEP do not sufficiently maintain end–to–end congestion control and rate control. Flow control and window clamping techniques, which are often used for this purpose, do not solve but work around the problem and undermine TCP's sliding window mechanism.

In this paper we propose the concept of Path Tail Emulation to overcome this weakness. A TCP flow is split at the gateway from the Internet into the mobile network. The mobile network is then hidden behind an emulated loss free link, the bandwidth and capacity of which correspond to the mobile network's properties.

Using Path Tail Emulation, congestion control and acknowledgement pacing is done exactly the same way as in pure wirebound networks, thus TCP can fully exploit the available network capacity without suffering from any adverse interaction with lower layers.

The behaviour of a mobile network behind a PEP using Path Tail Emulation is reduced to a network which is compliant to the network model used for TCP in wirebound networks and therefore can be handled with well known and well proven techniques.

1 Introduction

Internet access using wide area mobile netwoks like GPRS and UMTS (refered to as 'mobile networks' in this paper) is becoming increasingly popular. The typical situation is shown in figure 1. In this paper, we focus on the scenario where a sender in the Internet (FH) sends data to a mobile host (MH).

BS is the gateway between the Internet and the mobile network. This simplified view is sufficient for our discussion. [1]

In order to achieve sufficient throughput and to protect the sender from error loss, mobile networks employ reliable link layer protocols like the Radio Link Protocol (RLP) [BOW99, Bao96]. However, the use of RLP has adverse consequences for TCP

[1] Note that curent mobile networks hide user mobility from the Internet. Although MH is allowed to roam between different radio cells, MH will not change its IP address. In mobile networks this is typically achieved by 'micromobility' mechanisms on layer 2. Particularly BS is not related to any particular radio cell but will cover the whole mobile network. More detailed information on mobile networks can be found in [Wal01].

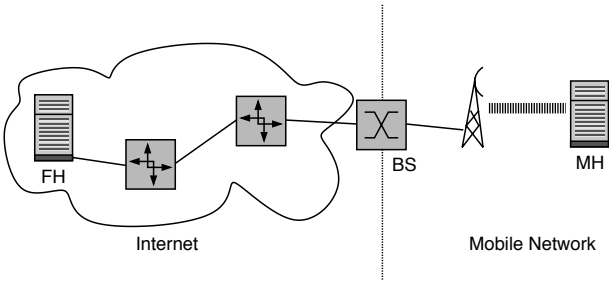

Figure 1: The system model

[BOW99], e.g. transport latencies in the mobile network suffer from large and abrupt fluctuations which make it difficult for the sender to achieve a proper, stable RTT estimate. In consequence, packet loss may be detected too late or falsely and insufficient throughput may result [Bao96, GL03]. This difficulty is overcome by connection splitting and performance enhancing proxies (PEP), which terminate the TCP flow and acknowledge TCP packets at BS and thus protect the flow from latency variation. BS in turn ensures proper packet delivery. To maintain proper end–to–end semantices for the TCP flow, BS does generate acknowledgements only if the TCP flow is in 'established' state. Particularly when the flow is shut down, acknowledgements are sent solely by MH. It is thus guaranteed that all data has been delivered to MH once a sender's 'close' has been properly acknowledged [CKCP03].

However, if TCP datagrams are acknowledged at BS, the end–to–end congestion control and TCP acknowledgement pacing are broken. If the mobile network happens to be the bottleneck, congestion at BS and therefore throughput degradation may occur [BOW99, Bao96, GL03]. The basic problem is that neither the mobile network's bandwidth nor the mobile network's capacity are signaled to the TCP sender in current PEP approaches. In fact, current PEP approaches work around this problem with flow control techniques and window clamping, which do not solve the problem but undermine TCP's sliding window mechanism. Refer to section 2 for a more detailed discussion.

In this paper, we propose the concept of Path Tail Emulation (PTE), which allows PEP techniques to provide the flow with the mobile network's bandwidth and capacity and therefore allows the sender to rely on proper congestion control and ACK pacing. A PEP using PTE does not simply hide the mobile network from the Internet, but covers only its disadvantegeous properties, whereas characteristics needed for proper pacing and congestion control are provided to the Internet, the flow and the TCP sender.

Providing the flow with the mobile network's bandwidth directly refers to the well proven principle of conservation [Jac88]: The mobile network accepts data at a rate it can successfully convey to the receiver. The average amount of data on the fly in the mobile network remains constant. Therefore, proper congestion control is achieved.

Providing the flow with the mobile network's capacity allows to extend TCP's sliding window technique on the whole path, including both the Internet and the mobile network. Thus, we avoid path underutilization, which may occur as a consequence of clamping techniques.

PTE fully decouples transport layer and link layer mechanisms. In addition, we do not abuse $awnd$ [2] as done e.g. by clamping techniques. Note that in TCP flow control

[2]We use the following abbrevations: $awnd$: Receiver's advertised window, $cwnd$: Congestion Window,

the sender is throttled, when the receiving *application* is unable to accept data from the socket and thus the sender must be stalled.

For the following discussion, we assume the system model shown in figure 1. The mobile link employs RLP and PEP techniques, hence its packet loss rate, packet corruption rate, packet duplication rate and packet reordering rate are negligible.

In order to provide the flow with the correct bandwidth, we need an estimate of the mobile network's average throughput to be provided by the mobile network's operator.[3] If a smoothened average is used, throughput fluctuations and short time disconnections can be dealt with locally by a PEP and hence are totally hidden from the flow.

Note, that we do not need an explicit estimate of the mobile network's capacity, different from the clamping approach presented in [CKCP03]. In pure wirebound TCP, $cwnd$ yields an estimate of the path's end–to–end capacity. Using the PTE approach presented in this paper, $cwnd$ will continue to do so even in a split connection or in presence of a PEP.

The remainder of this paper is organized as follows. In section 2 we discuss some related work. Our approach is described in section 3. Some first simulation results are presented in section 4. Section 5 summarizes our work and gives an overview of our next steps.

2 Related Work

Current split connection approaches and performance enhancing proxies usually employ the receiver advertised window ($awnd$) to achieve proper congestion control. [BB95, SRWB01] for example rely upon TCP's flow control mechanism. A modified version of flow control are *clamping approaches*. These approaches 'clamp' the sender window by an appropriate value. In [CKCP03] $awnd$ is set to the available buffer space in the mobile network in order to avoid congestion at BS. The idea of [AHM03] is to maintain a constant buffer queue at the path bottleneck by restricting the sender's window using $awnd$.

In our opinion, there are some basic shortcomings in using $awnd$ for congestion control.

1. $awnd$ is typically not constant but may suffer from large fluctuation, especially when $awnd$ is used for clamping. In [CKCP03] $awnd$ is set to the mobile network's free capacity, which heavily depends on wireless conditions, because link layer retransmissions occupy buffer space as well as line capacity. These fluctations are imposed on $swnd$ and thus practically undermine the sliding window mechanism used in TCP. Note that $cwnd$ is an estimate of the flow's fair share of the path capacity and thus should remain rather constant, apart from probing. If a receiver's buffer, and therefore $awnd$, is sufficiently large, $swnd$ equals $cwnd$.
2. Another difficulty in [AHM03] is that its setting of $awnd$ violates the requirements for TCP receivers as specified by the IETF [Bra89]. Particularly, [Bra89] specifies the correct window update behaviour for $awnd$ to avoid the 'Silly Window Syndrome'.
3. $awnd$ may be zero, e.g. in M-TCP[BS97] this is used to stall the sender in case of roaming, and therefore a window update becomes necessary. Window updates are sent unreliably and TCP senders follow a timer backoff scheme when

$swnd$: Sender's window, i.e. the minimum of $awnd$ and $cwnd$

[3] Mobile networks require quite a lot of monitoring and compensation of path disturbances, e.g. phase shift due to multipath fading. Therefore it should be possible to provide a througput estimate with reasonable effort.

window updates must be polled. In consequence, unwanted pauses in the flow may occur.

4. If the flow experiences the bottleneck in the *Internet*, there should be no problem at all. A flow from FH to MH arrives at BS with a rate that the mobile network can carry. However, clamping approaches like [CKCP03] restrict $swnd$ by setting $awnd$ to the mobile network's capacity. In flow control approaches like [BB95, SRWB01], $awnd$ is set to the available buffer space in the receiving socket. Neither approach considers the capacity available in the Internet. This may result in the following situations:
   (a) $cwnd$ is less than or equal to $awnd$. In this case, the TCP flow is left alone. Because by assumption the TCP flow is already restricted to the proper rate, this is exactly what we want.
   (b) $awnd$ is less than $cwnd$. In that case, a TCP flow's rate is decreased by BS although its rate may be carried by the mobile network. In that case, the mechanism may happen to be malicious.

   Note that $cwnd$ can exceed $awnd$ particularly in cases with a long Internet path with large path capacity. Typically, this is not a lab situation where the "Internet" may be represented by one or two LAN segments with only little capacity. However, in practical situations, e.g. a UMTS subscriber in Berlin wants to download some material from a server in the United States, the wirebound part of the path may have much more capacity than the mobile one.

5. Another problem occurs, if the flow experiences the bottleneck in the *mobile network*. Let us clearly point out what we try to achieve: The sender shall use its fair share of path capacity and thus send with its ideal rate. The relation between rate and $cwnd$ can be roughly described by the well known formula $rate = \frac{cwnd}{rtt}$. Hence, to have the sender send at the proper rate, the sender's window must be set, or clamped respectively, to a value which respects the rtt as seen by the sender, which is difficult and not supported in current TCP flavours.

In summary, clamping and flow control techniques interfere with TCP's sliding window mechanism because the path capacity located in the Internet is not considered.

Some approaches stop the sender during handover [BS97] or in case of a broken mobile network [GMPG00]. In both cases, the sender is blocked by setting $awnd$ to zero, which leads to the aforementioned difficulty.

Freeze TCP [GMPG00] attempts to signal bad conditions on the mobile network to the sender using $awnd$; in other approaches network conditions are signaled by BS, using flags or duplicate acknowledgements. One example is the FDA scheme presented in [HY01]. A basic difficulty in these approaches is to timely slow down and accelerate the sender. If e.g. the sender is slowed down too early, an unwanted pause in the flow would result. If the sender is re-accelerated too early, packets may reach the mobile network although the bad condition is not yet overcome.

With PTE, there is no slow down at all. Instead, varying network conditions are hidden by a PEP, which in turn emulates a link with a given bandwidth to the Internet.

The 'ACK Regulator' presented in [CR02] attempts to overcome problems caused by delay variation in mobile networks by regulation of ACK datagrams sent from MH to FH. The key idea is to delay ACK packets and thus to slow down the sender in case of imminent buffer overflow at BS, when $cwnd$ has not yet reached a certain threshold $QueueLim$. Therefore, it is necessary to provide an estimate for $cwnd$ at BS and to define an appropriate threshold $QueueLim$. In general, it is difficult for BS to obtain an estimate for $cwnd$. The algorithm presented in [CR02] works fine for the scenario used in [CR02]. However, this algorithm will not hold for the general case. Particularly, any drop in the Internet, e.g. due to crosstraffic, traffic bursts or queue management mechanisms as RED would cause the presented mechanism to fail.

Note, that the PTE approach does *not* make any assumptions on the Internet. PTE does *not* attempt to estimate state variables used at the TCP sender or other state variables in the Internet. Instead, PTE provides the Internet with information which can be reliably obtained from the mobile network's operator.

The very strength of PTE is that PTE does not add any mechanisms to the Internet or to TCP, but mobile networks are fit into TCP's system model and thus can be successfully treated by current TCP control mechanisms as well as by future ones.

3 Path Tail Emulation

The basic goal in Path Tail Emulation is to provide the Internet with the bandwidth and path capacity of the mobile network.

Let us briefly discuss a TCP datagram traveling from FH to MH and the corresponding ACK datagam in an idealized situation. Consider a loss free network segment between BS and MH with a certain bandwidth bw and constant propagation latency as depicted in figure 2. For simplicity, we consider a unidirectional flow from FH to MH.

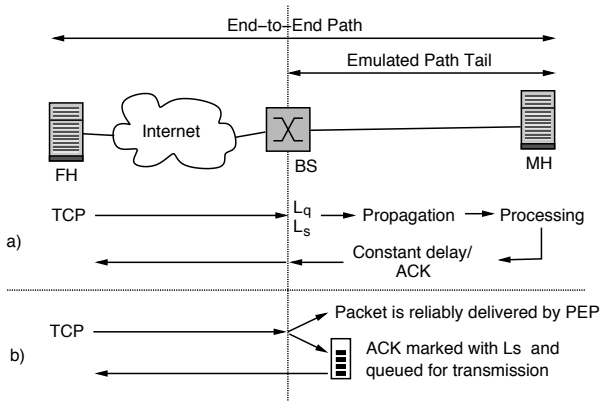

Figure 2: Latencies of a TCP packet. The mobile network is modeled as loss free link with a bandwidth equal to the mobile network's estimated througput in figure (a). Figure (b) shows the Path Tail Emulation algorithm. The PEP ensures reliable transmission.

A packet traveling from FH to MH reaches BS at time T_{TCP} and experiences the queueing latency L_q and the serialization latency L_s. After a constant propagation delay it arrives at MH. We assume the processing latency at MH to be constant (in fact, it is negligible in many cases). After the processing latency MH issues an ACK datagram which is then sent back to FH via BS. In unidirectional flows, ACK datagrams carry no data and thus have a constant length and experience constant serialization latencies at MH. The propagation latency from MH to BS is constant. The ACK datagram reaches BS at time $T_{ACK} = T_{TCP} + L_q + L_s + c$, where c summarizes the aforementioned constant latencies. L_s depends on the actual packet length and the bandwidth bw.

The key idea of Path Tail Emulation is to emulate exactly this behaviour at the real BS: If a TCP datagram arrives at BS, BS checks whether the datagram can be accepted for delivery. If the datagram cannot be accepted due to buffer overflow, the datagram is silently discarded. In consequence, $cwnd$ will be correctly decreased by normal congestion handling. If the datagram is accepted for delivery, BS sends the

appropriate acknowledgement to FH after a delay $D = L_q + L_s$. This results in correct acknowledgement pacing and increase of $cwnd$.

D does not contain the constant c, because c does not affect the temporal gaps between ACK datagrams and therefore has no meaning in acknowledgement pacing. c corresponds to a constant delay and therefore to a constant part of the path capacity. Hence, c is taken into account implicitly by the sender in $cwnd$, which is an estimate of the path capacity. The serialization latency L_s is calculated from the *packet length* and the *throughput estimate provided by the network operator*. The queueing delay L_q is calculated implicitly by an ACK queue maintained on BS. When BS accepts a TCP datagram, it calculates L_s and appends the ACK datagram marked with L_s to the ACK queue. The ACK queue is served FIFO. When an ACK datagram is taken from the ACK queue the serving process waits the serialization delay corresponding to this ACK. Afterwards, the serving process sends the ACK datagram and continues serving the queue.

4 Simulation

For a first proof of concept, we implemented our algorithm with the network simulator NS2 [ns2]. We simulated the mobile network as a loss free link without packet duplication and packet reordering but with large and varying latencies, similar to current GPRS and UMTS networks. Our simulation topology is similar to figure 1. The 'Internet' is modeled as a single link connecting FH and BS with 100 KBit/s bandwidth and 50 ms latency. We chose these parameters to model some reasonable Internet connection as it may appear to an Internet user. For the mobile link we chose a link bandwidth, i.e. the parameter bw, of 1 MBit/s and a propagation delay of 1 ms.

Link layer recovery and roaming was modeled with a varying recovery with an average of 0.5 s. In the implementation, this latency was added to the serialization delay in order to leave the packet order untouched. We used a Pareto distribution to model varying latencies in the mobile link, because this distribution exhibits values which are typically rather small but sometimes expose large outliers.

We implemented a trivial PEP to hide varying latencies from the Internet using a simple buffer mechanism. Newly arrived packets are stored in a buffer which is served by the mobile network. If the buffer is occupied, incoming packets are discarded. Our PEP employs the PTE algorithm as presented in this paper. The serialization delay L_s was calculated according to the aformenetioned link bandwidth 1 MBit/s. The average recovery latency 0.5 s was added to L_s. This is equivalent to the calculation in section 3 where link bandwidth and average recovery delay appear as one parameter, i.e. the throughput estimate. The buffer space was set to 1000 packets.

For a first proof of concept, we compared TCP flows with and without proxy. In the first situation, TCP flows were confronted with the varying latencies which result from RLP, roaming etc. In the second situation, TCP flows were protected from the mobile network by our proxy and the PTE mechanism.

Figure 3 shows a typical result. The x-axis shows the simulation time in seconds, the y-axis shows the amount of data received by the *application*. In all simulations, we transfered a fixed amount of data (2 MByte) and thus could compare the duration of the copy process.

In the diagram, we have a straight line which describes the bandwidth of the mobile link, which is the bottleneck in our simulations. Thus, it describes the maximum possible *throughput* of the mobile network.

The curve with circles shows the amount of received data with PTE switched on. The curve with triangles shows the amount of received data without PTE. Because the amount of data transfered is identical, the transfer durations can be compared. In the simulation depicted in figure 3 the transfer without a proxy enabled needs about 230

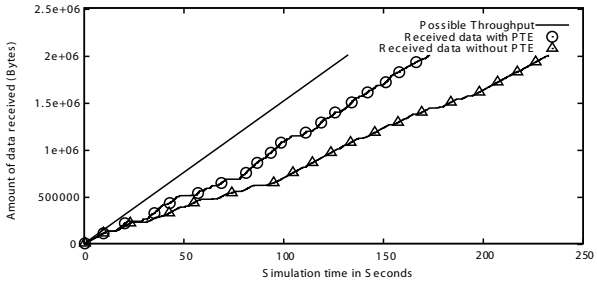

Figure 3: TCP flow with and without PTE

seconds to complete, whereas with proxy enabled the transfer only needs 170 seconds.

In our future work, we will compare the PTE mechanism to other approaches for congestion control and rate control. The most important ones are flow control and clamping approaches. Unfortunately, the NS2 does not yet support dynamic $awnd$, update window polling and proper avoidance of the Silly Window Syndrome for TCP receivers. It is thus necessary to add these mechanisms before we can conduct comparative simulations.

5 Conclusion and Future Work

We presented the concept of Path Tail Emulation which allows the seamless application of the TCP system model to nodes attached to the Internet by mobile networks, connection splitting and performance enhancing proxies.

However, this work is far from being complete. So, the next paragraphs will give an overview of our next steps.

1. We will conduct comparative simulations to other approaches. Therefore we will add full $awnd$ flow control and the window update mechanism to NS2 in order to properly simulate flow control and clamping approaches.
2. We will validate our approach and demonstrate its effectiveness with a generic model of mobile networks. In this short paper, we used a simple approximation based upon a Pareto-distribution. For our future work, we will use a model based upon the work presented by Gang Bao [Bao96]. The main idea is to use a generic RLP implementation in conjunction with varying radio block error conditions.
3. A main objective of path tail emulation is to correctly provide the Internet with the correct bandwidth for a flow. Basically, the bandwidth is obtained by the mobile network's operator. However, a number of different cases must be considered. For space limitations, we can only give some examples here.
   (a) *BS serves one broadcast cell.* In this case, all TCP flows directed to this broadcast cell share the available bandwidth. bw is set to the cumulative bandwidth for this cell. This situation is equivalent to other shared media networks like Ethernet. Thus, the flows are allocated a fair share of ressources as a consequence of TCP congestion control.
   (b) *BS serves different broadcast cells.* In this case, BS should implement the path tail emulation algorithm for each cell individually to achieve a fair share of ressources within each cell.

(c) *The mobile network allows QoS profiles*. Some mobile networks, like UMTS, allow fixed bandwidth allocations for individual flows. Although the Internet itself is typically 'best effort', path tail emulation allows allocating a fixed bandwidth to a flow. This can be achieved by implementing the emulation algorithm individually for each flow.

References

[AHM03] Lachlan L. H. Andrew, Stephen V. Hanly, and Rami G. Mukhtar. CLAMP: A System to Enhance the Performance of Wireless Access Networks. In *Proceedings of the GLOBECOM 2003*, December 2003.

[Bao96] G. Bao. Performance Evaluation of TCP/RLP Protocol Stack over CDMA Wireless Link. In *Proceedings of the IEEE ICCT*, pages 710–713, 1996.

[BB95] Ajay Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. *15th International Conference on Distributed Computing Systems*, 1995.

[BOW99] Y. Bai, A. T. Ogielski, and G. Wu. Interactions of TCP and Radio Link ARQ Protocol. In *Proceetings IEEE VTC-Fall*, pages 1710–1714, Amsterdam, The Netherlands, September 1999.

[Bra89] R. Braden. Requirements for Internet Hosts – Communication Layers. IETF RFC 1122, October 1989.

[BS97] Kevin Brown and Suresh Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communication Review*, 27(5), 1997.

[CKCP03] Rajiv Chakravorty, Sachin Katti, Jon Crowcroft, and Ian Pratt. Flow Aggregation for Enhanced TCP over Wide Area Wireless. In *Proceedings of IEEE INFOCOM 2003*, 2003.

[CR02] Mun Choon Chan and Ramachandran Ramjee. TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation. In *Proceedings of the ACM MobiCom*, pages 71–82, Atlanta, Georgia, USA, 2002.

[GL03] Andrei Gurtov and Reiner Ludwig. Responding to Spurious Timeouts in TCP. In *Proceedings of IEEE Infocom 2003*, March 2003.

[GMPG00] Tom Goff, Jim Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments. In *Proceedings of the IEEE INFOCOM 2000*, pages 1537–1545, 2000.

[HY01] 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*, 2001.

[Jac88] V. Jacobson. Congestion Avoidance and Control. *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, 18, 4:314–329, 1988.

[ns2] The Network Simulator ns-2. http://www.isi.edu.nsnam/ns. Version 2.26.

[SRWB01] M. Schläger, B. Rathke, A. Wolisz, and S. Bodenstein. Advocating a Remote Socket Architecture for Internet Access using Wireless LANs. *Mobile Networks and Applications, Special Issue on Wireless Internet and Intranet Access*, 6(1):23–42, 2001.

[Wal01] B. Walke. *Mobile Radio Networks, Networking and Protocols*. John Wiley and Sons, Chichester, 2nd edition, 2001.