On TCP Performance Enhancing Proxies in a Wireless Environment

Milosh Ivanovich and Philip W. Bickerdike, Telstra Corporation

Jonathan C. Li, University of Melbourne

Jonathan C. Li was affiliated with Telstra Corporation for the duration of this work.

ABSTRACT

TCP performance enhancement in wireless access networks is an important ongoing area of research. It is known that the hostile nature of the wireless channel and the mobile nature of wireless users interact adversely with standard TCP congestion control mechanisms [1], causing a drastic reduction in throughput. This article surveys a selection of different approaches to managing TCP performance over wireless links, and presents the results of simulation and field trial results of a novel TCP performance enhancing proxy over diverse cellular radio access technologies based on the GSM, cdma2000, and UMTS standards. The proposed TRL TCP performance enhancing proxy has the advantages of being completely transparent to both TCP endpoints and tunable to different access technologies, and frequently achieves the maximum throughput available from any of the studied radio access technologies.

INTRODUCTION

In this article we begin by reviewing some key concepts that dictate the behavior of TCP. Next, we give examples of different mechanisms in the cross-layer design literature [1, 2] that have been considered for enhancing the performance of, or replacing, TCP over wireless links. This review is representative of various TCP enhancement concepts that exist, but is by no means exhaustive. We then describe a novel algorithm we have developed, selectively making use of some of the most promising ideas reviewed. Finally, we present simulation and field trial results of the new algorithm.

KEY TCP CONCEPTS

Historically, TCP's main area of application has been in wired networks, where user traffic congestion is the main cause of packet loss, and errors due to the transmission media are rare. Subsequently, TCP behavior has been optimized to deal with these issues. However, when the transmission medium is a wireless link, such "traditional" TCP behavior is inappropriate, because radio-induced packet errors overtake congestion as the dominant source of packet loss.

Consider Fig. 1, where the vertical axis represents the amount of unacknowledged data (called the congestion window or CWND) the TCP sender is permitted to transmit to the receiver, and the horizontal axis is time. The dotted red line represents the optimal bandwidth delay product for the transmission link — the amount of sender unacknowledged data required to fully utilize the TCP "pipe." For this reason, we seek to keep the blue line (effectively a measure of achieved throughput) close to the red dotted line (the maximum possible link throughput). Figure 1 shows TCP's normal reaction to a packet loss (signaled by triple duplicate acknowledgments or DUPACKs) and to a retransmission timeout; in the former case, the congestion window is halved, while in the latter it is reduced to one segment. In a wireless network with packet losses due to random radio events, such substantial throughput reduction is often unnecessary.

Assuming that the reader has a basic level of understanding of TCP protocol fundamentals, we turn our attention to reviewing recent performance enhancement mechanisms proposed for TCP over wireless networks. We refer to NewReno, the most widely deployed TCP variant [3] in the Internet, as the de facto standard.

TCP SELECTIVE ACKNOWLEDGMENTS OPTION

The selective acknowledgment (SACK) option [4] available in the TCP specification modifies acknowledgments (ACKs) sent by the receiver back to the sender by including information on exactly which segments have been received and which segments need to be retransmitted. It is up to the sender's TCP implementation to make the best use of this additional data. SACK is commonly used in most modern TCP implementations.

MODIFIED TCP APPROACHES

These approaches rely on the modification of the sender and/or receiver TCP implementations. Such approaches are typically undesirable, as deployment is hampered by inertia of the large population of existing TCP versions in the global Internet, as well as potential incompatibilities between new and old versions (e.g., fairness).

TCP Vegas [5] introduces a different timeout mechanism, a change to congestion avoidance that attempts to control the occupancy of buffers for the connection in the network, and a modified slow-start mechanism. Using delay measurements, Vegas attempts to eliminate periodic self-induced segment losses caused in standard NewReno TCP. TCP FAST [6] is a more recent variant with similarities to Vegas, although thus far it has been studied mainly in the context of very high-speed optical links.

TCP forward acknowledgment (FACK) [7] separates congestion control algorithms from data recovery algorithms. It operates with the TCP SACK option to keep an explicit measure of the total outstanding (unacknowledged) data in the connection. Since FACK more accurately controls the outstanding data in the network, it is less bursty and recovers from heavy loss more efficiently than standard TCP NewReno.

TCP Westwood [8] exploits two basic concepts: end-to-end estimation of available bandwidth, and the way such an estimate is used to set the CWND. It properly averages the rate of returning ACKs at the TCP sender to compute a per-connection bandwidth estimate (BWE). Upon reception of a triple DUPACK, the CWND and slow start threshold are set according to the measured rate of the connection (BWE). Upon a retransmission timeout, the CWND is set to one. Only modification to the TCP sender is required for implementation. The scheme has been shown to perform well in the presence of segment loss caused by link errors, which are typical in wireless environments.

Similar to TCP Westwood, TCP Jersey [9] employs a more refined bandwidth estimation algorithm, as well as explicit congestion notification (ECN) support, to make better decisions in controlling TCP sender behavior.

Split Connection Approaches

This section describes approaches that introduce an intermediary between sender and receiver to mitigate link errors. This category can be divided into approaches that are transparent or nontransparent. In nontransparent approaches the intermediary establishes separate connections with both the sender and receiver. A nontransparent approach violates the end-to-end semantics of the TCP connection and requires that both end systems be aware of the intervening device. Application layer proxies for TCP-based applications such as HTTP are a common example of such an implementation. Conversely, a transparent approach is one that requires no modifications to TCP hosts at either end of the connection.

TCP Snoop [10] is an example of such a transparent TCP-aware link layer protocol designed to improve the performance of TCP, particularly over hybrid networks of wired and single-hop wireless links. It requires modification to the base station node of the wireless network through the deployment of a "snoop agent"; however, the TCP endpoints do not need to be aware of the agent, yielding transparency. A buffer is maintained of all unacknowledged TCP segments transmitted across the wireless link to facilitate retransmissions of lost segments. The intention is to suppress the mobile host's DUPACKs (corresponding to wireless link losses) in order to prevent unnecessary invocations of congestion control mechanisms at the TCP sender.

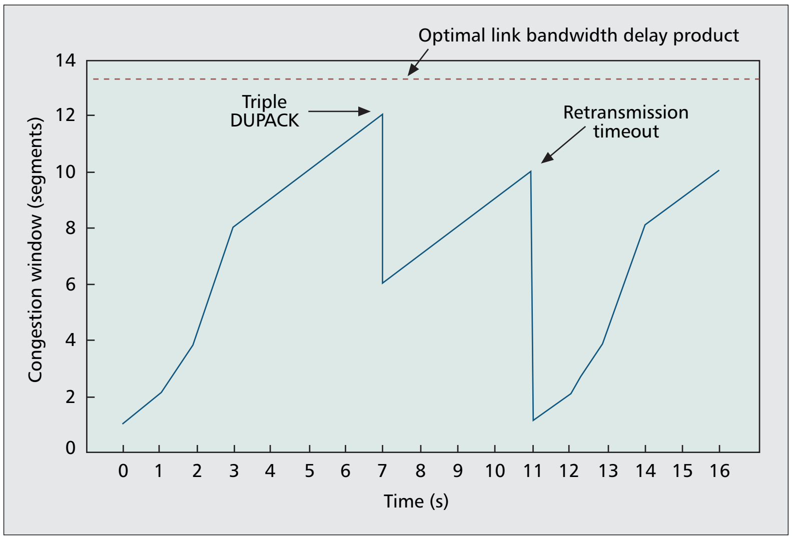


Figure 1. *Standard TCP behavior for random packet loss and retransmission timeout.*

M-TCP [11] may be thought of as a "partially" transparent split connection scheme, because it modifies TCP on the mobile host to respond to notifications of wireless link connectivity. M-TCP introduces an intermediate node, the Supervisor Host (SH); however, separate split connections with the TCP endpoints are still not required, so we have a form of transparency as per the definition above. The SH ensures that the TCP sender does not invoke congestion control mechanisms during wireless link loss by sending an ACK containing a TCP receiver window size update of zero bytes that forces the sender into "persist mode." In this state the sender will not suffer from retransmit timeouts that cause the retransmission timer to back off or reduce its CWND. When the receiver resumes normal operation, the SH transmits a non-zero receiver window advertisement to the sender, which resumes transmitting at full speed. M-TCP relies on the SH, instead of the CWND, to regulate transmission to the receiver.

TCP Replacement Approaches

Replacement approaches emulate TCP, but replace the actual transport layer protocol over the wireless medium with proprietary offerings. Such approaches do away with the limitations of TCP and provide additional functionality such as data compression at the cost of no longer being freely available in the public domain. These solutions are aimed at the telecommunications carrier and service provider market, having nontrivial purchase and implementation costs due to the proprietary nature of the software, and typically require a client installation on the user device.

VTP [12] is a proprietary protocol developed by Fourelle Systems and implemented by Venturi Wireless as part of a suite of tools for enhancing wireless TCP performance, including protocol optimization, intelligent content com-

The TRL-PEP would be located as close to the wireless TCP receiver as possible, while still being far enough from the end-point that it may optimize traffic destined for a range of users. This is to ensure that all traffic destined for the wireless segment of the network passes through the TRL-PEP.

pression, and high-speed caching. The protocol is UDP-based and completely replaces TCP on the wireless link. Exact implementation details are unknown due to the proprietary nature of VTP, but it claims to add reliability to UDP while maintaining a standard UDP framework to ensure interoperability. The architecture requires deployment of a server in the network and a software client on the mobile host.

Developed by Flash Networks, Wireless Boosted Session Transport (WBST) [13] is a component of Flash Networks' NettGain product. Similar to VTP, it is designed as a reliable transport layer protocol and requires the replacement of TCP on the wireless device and deployment of a NettGain server. WBST applies bandwidth estimation techniques and rate-based control to increase the bandwidth utilization of the wireless link. The WBST sender continually holds a notion of transmission rate and monitors the ACK interarrival time. To accommodate changes observed in this interarrival time (which indicate a change in link bandwidth), the interpacket transmission delay is adjusted accordingly. The intention is to utilize all of the available bandwidth without causing congestion.

The Proposed Algorithm

Most of these approaches for enhancing TCP in wireless environments have both desirable and not-so-desirable algorithmic and implementation features. We therefore developed a new TCP performance enhancing proxy, TRL-PEP, that makes use of the most desirable functionality of the reviewed algorithms as well as additional novel enhancements.

The TRL-PEP is designed as a proof-of-concept prototype, specifically for TCP optimization in a wireless environment, catering for all applications that run over TCP. It is a transparent split connection approach that decouples the sender-to-PEP and PEP-to-receiver TCP control loops while maintaining the end-to-end TCP semantics. The TRL-PEP has a downlink focus (server to user device), aiming to maximize link utilization by maintaining a full buffer of data from the sender and using modified TCP behavior to transmit to the wireless receiver.

The main features of the TRL-PEP include:

- Clientless solution with transparency to TCP connection endpoints, requiring no modification to, or intervention on the part of, sender or receiver.
- Combining into one solution different state-of-the-art ideas in the field:
  - M-TCP — use of TCP Persist mode and Zero Window Advertisements to pause the TCP sender while maintaining state variables (TCP sender control)
  - TCP Snoop — use of a buffer at an intermediate node and local retransmissions over the wireless link to the receiver (TCP receiver control)
  - TCP FACK — use of the fact that the presence of DUPACKs indicates that more data can be sent to the receiver (TCP receiver control)
- Novel algorithms including:
  - Efficient use of radio resources through aggressive link utilization and loss recovery mechanisms
  - Intelligent detection of loss coupled with SACK mechanisms where available
  - Intelligent inference and mitigation of the connection state encompassing: lost retransmissions, spurious timeouts, congestion and state prediction of TCP hosts

Some of these novel algorithms are detailed below.

Network Location

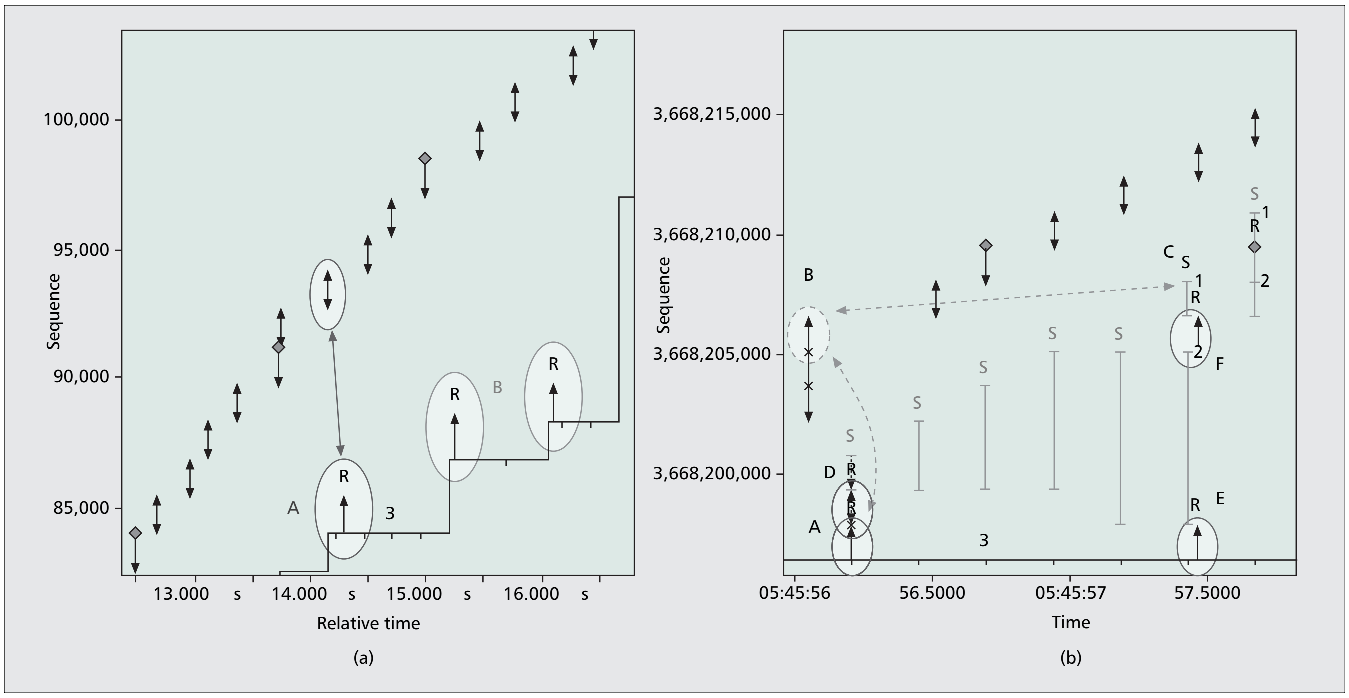
The TRL-PEP would be located as close to the wireless TCP receiver as possible, while still being far enough from the end-point that it may optimize traffic destined for a range of users. This is to ensure that all traffic destined for the wireless segment of the network passes through the TRL-PEP.

The TRL-PEP works on the simple principle: that it should attempt to fully utilize the wireless link to a receiver at all possible times. Network topology allows certain assumptions with regard to the application of this principle:

- Most TCP segment loss and delay variation experienced by the receiver is due to wireless channel errors rather than network congestion, which would only occur in the unlikely scenario of multiple users in the same radio cell simultaneously:
  - Obtaining excellent radio conditions
  - Trying to fill the wireless link with large amounts of traffic
- Although TRL-PEP's main focus is mitigation of wireless channel errors, it still needs a basic congestion control capability to protect against the unlikely event of true congestion-based losses. We achieve this through a simple algorithm triggering a CWND reduction of X segments whenever Y consecutive segments are lost (whether identified as lost by sequence number inference or timeout). X and Y are dynamically configurable variables. As in standard TCP's congestion avoidance phase, the CWND then grows back by one segment per round-trip time (RTT) of loss-free operation.
- TCP segments are assumed to be delivered in order to the receivers as there are no alternate network paths from the TRL-PEP to the wireless receivers.

Aggressive Link Utilization

Once a connection is established between a sender and receiver through the TRL-PEP, data is requested by the TRL-PEP from the sender by actively spoofing ACKs to the sender until the connection buffer at the TRL-PEP has been filled. The advertised window (Awin) parameter in the spoofed ACKs is reduced in proportion to the available buffer space, to the point at which the TCP sender will stop sending data until the Awin is increased again. This is a standard TCP mechanism to control the flow of data from the sender. If the Awin is set to zero, the sender will enter the TCP standard Persist mode, which effectively pauses the sender while maintaining its state variables.



■ **Figure 2.** *TRL-PEP algorithmic improvements: a) partial ACK behavior; b) intelligent use of selective ACKs.*

TCP's conservative slow-start behavior is less appropriate in the typically high-bandwidth high-delay connection between a sender and receiver in today's broadband wireless networks. We observe a "thick long pipe" in cdma2000 and UMTS standards, compared to the "thick short pipe" analogy of wireline accesses such as asynchronous digital subscriber line (ADSL) or cable. Thus, the TRL-PEP transmits data to the receiver at the highest rate supportable by the wireless network. In the simulation studies reported here, this "feed rate" for the TRL-PEP-to-receiver link was separately preconfigured based on "typical" values of bandwidth and delay for each of the wireless access technologies under study. Although beyond the scope of the proof-of-concept prototype, a relatively simple extension could:

- Facilitate dynamic estimation of the optimal feed rate (e.g., via RTT measurements or ACK counting)
- Simultaneously support connections to receivers on different wireless access networks by separately estimating and managing the feed rate on a per connection basis

ADDITIONAL DATA TRANSMISSION ON DUPACKS

When the TRL-PEP encounters a DUPACK, it allows extra data to be transmitted. The rationale for this, similar to that used in standard TCP NewReno's Fast Recovery phase, is the reception of a DUPACK infers that a packet has been received correctly by the receiver, so there is available capacity in the link for more data to be transmitted. This extra data pushed into the link, and the resultant DUPACKs also act as a probing mechanism to determine if the link has been disconnected or overutilized.

RETRANSMISSION ON PARTIAL ACKNOWLEDGMENTS

The TRL-PEP has the capacity to recover from multiple packet losses within the same window of transmitted data. This behavior is specific to the case where SACK information from the receiver is not available, and is governed by the following algorithm:

- Upon retransmission from a DUPACK, a variable *dupack_target* is set to the highest sequence number sent to the receiver thus far (shown as A in Fig. 2a). Note that there is no requirement that the packet numbered *dupack_target* has been successfully received.
- When a new ACK numbered less than *dupack_target* is received (known as a partial ACK), the packet with the corresponding sequence number is immediately retransmitted (shown as B in Fig. 2a).
- Repeat step 2, until a new ACK numbered greater than *dupack_target* is received, at which point we return to normal TCP operation.

This algorithm is based on the premise that at the time partial ACK B is received, the TRL-PEP has knowledge that a higher sequence numbered packet has been transmitted as indicated by the presence of a DUPACK. Thus, the received partial ACK is an indication that the next-in-sequence packet was lost. This also presumes that packets (and ACKs) arrive in order.

INTELLIGENT USE OF SELECTIVE ACKNOWLEDGMENTS

The TRL-PEP makes use of SACK data to retransmit only necessary packets to the receiver. In order to respond immediately to many concurrent retransmissions of data based on

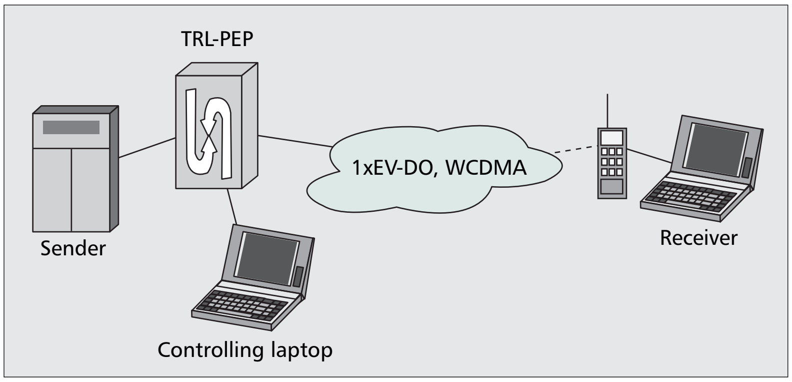


Figure 3. *TRL-PEP test network topology.*

SACK information and quickly detect subsequent losses of these retransmissions, the TRL-PEP maintains two arrays of sequence numbers: *sack_seq[]* and *sack_target[]*. The *sack_seq[]* array records the sequence number of the segment that has been retransmitted in response to SACK, while *sack_target[]* records the corresponding ACK number that ought to be received before the retransmitted data (use of these arrays is illustrated in an example below). Thus, it is possible to repeat the retransmission without waiting for a retransmission timeout. In this way the proposed algorithm mitigates the loss of multiple transmitted and retransmitted segments within a single RTT period, unlike standard TCP. An example of this mechanism in operation is shown in Fig. 2b with a time sequence graph of a TCP connection with the SACK option enabled. The vertical line segments marked S depict information from the receiver informing the sender (in our case, the TRL-PEP node in the middle of our end-to-end path) of exactly which packets have not been received correctly.

Consequently, in Fig. 2b the packets marked A, D, and F are all retransmitted as soon as the TRL-PEP receives enough information to determine that those packets have not been received. In this example, prior to packet A being retransmitted, *sack_target[0]* is set to the highest sequence number sent, as shown by packet B. Since it is the lowest outstanding segment, the sequence number of packet A is recorded in *sack_seq[0]* and then retransmitted. As shown in the figure, the same SACK information used in deciding to resend A is used to indicate that the next sequence number to be resent is that of packet D. This, then, is the lowest outstanding segment that has not already been retransmitted, so it is recorded in the next available position in *sack_seq[1]*. *sack_target[1]* is also set accordingly (i.e., to equal the sequence number of B), and packet D is retransmitted. SACK information continues to arrive with the DUPACKs, and new data is sent accordingly, until the SACK information arrives at C. The highest ACK number indicated by the SACK information at C becomes greater than *sack_target[0]*, while the ACK number is still less than *sack_seq[0]*. Assuming as before that packets (and ACKs) arrive in order, the TRL-PEP infers that retransmission A of the segment has been lost and retransmits *sack_seq[0]*, as shown by packet E (*sack_target[0]* is reset to the new value of highest sequence number sent). C contains two pieces of additional information:

- Packet D has been successfully received, so it is removed from the arrays.
- Another packet is lost, as indicated by the SACK sequence number "gap" at F. The corresponding sequence number is recorded as the new *sack_seq[1]*, *sack_target[1]* is set accordingly, and packet F is retransmitted.

TCP State Prediction

The TRL-PEP maintains the TCP sender and receiver states. This is achieved by observing the flags of packets received from the hosts and making use of knowledge of the standard TCP states and state transitions, as described in RFC 793. For example, at the beginning of a connection, should we receive a SYN flag, we know that the sender of this packet would be in the SYN_SENT state, and later, when we receive a SYN-ACK from the receiver, that the receiver would be in the SYN_RCVD state.

This knowledge of the TCP end-host states is essential for maintaining integrity of the connection as it allows us to manipulate the hosts for optimal data transfer, and gracefully establish and terminate connections.

Simulation Results

A simulation of the TRL-PEP algorithms was developed in the ns-2 discrete event network simulator.

For the development of the TRL-PEP we installed ns-2 (version 2.26) on a Linux Redhat 6.2 (Kernel 2.4.18) platform. To this standard installation we added custom-built simulations of General Packet Radio System (GPRS) (GSM) and 1xRTT (cdma2000) wireless links. ns-2 contains a range of TCP implementations we were able to enhance to include features such as dynamic window advertisement and TCP persist mode. We also enhanced the trace file format to include additional information such as advertised window, SACK, and other TCP options for use with a standard TCP analysis tool.

Tests were performed using simulated GPRS and 1xRTT links for varying uniformly distributed packet error rates, packet jitter, and different TCP implementations in the network topology depicted in Fig. 3. The standard TCP implementation is NewReno with SACK options enabled. Figure 4 shows the typical performance improvement using simulated 1xRTT in the presence of a bandwidth oscillation effect particular to the technology [14]. The TRL-PEP mitigates this behavior to clearly outperform other TCP implementations at different packet error rates. Typical performance improvement in GPRS was similar but not as pronounced.

In any comparison between the TRL-PEP and other schemes, it should be reinforced that the former requires no change to sender or receiver side TCP implementations; nor does it require the installation of any software clients on the user device. For a network provider, these represent significant advantages in favor of the

TRL-PEP. The simulation was developed as a precursor to the proof-of-concept prototype. Simulation code was sufficiently modular to be directly used in implementation of the PC-based prototype, which was subsequently put into a real network for a live trial. The trial results are detailed in the next section.

Live Trial Results

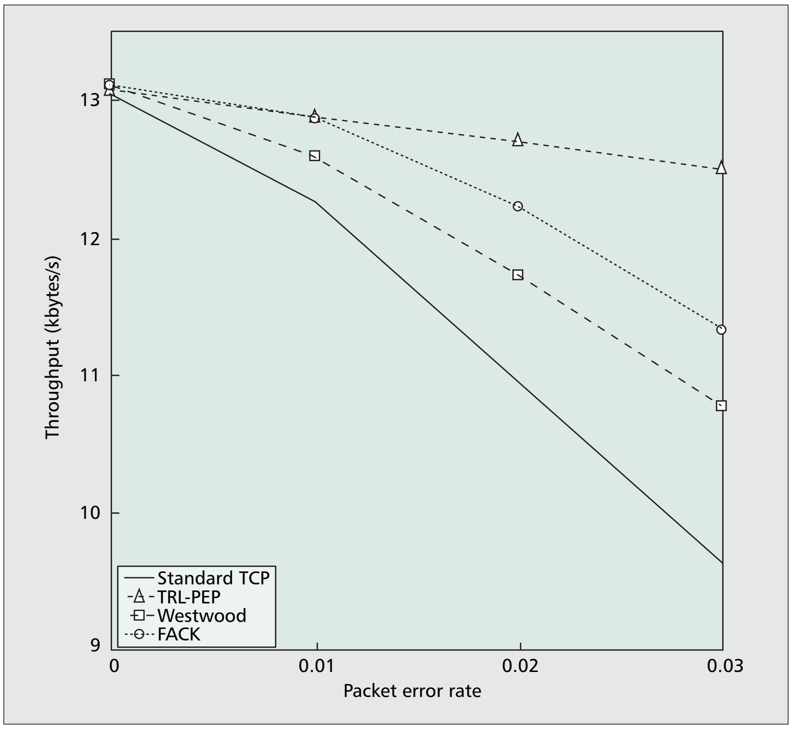
The prototype was implemented on a *Knoppix 3.4* (Kernel 2.4.26) Linux operating system due to its freely available associated tools and libraries, which were required for development of the TRL-PEP. All code was written in C++ and compiled using GNU Compiler Collection (GCC) version 3.3.3. TRL-PEP simulation code was reused with little or no modification in the prototype implementation.

The hardware used for the prototype was a standard desktop PC (Celeron 2.0 GHz) with 256 Mbytes RAM. The system had three 100 Mb/s Ethernet interfaces, a wireless facing interface, an Internet facing interface, and a controller interface. The latter was used for remote administration and monitoring, and had no functional purpose for the operation of the TRL-PEP. The topology from Fig. 3 was used to test the TRL-PEP in controlled environment conditions over the wideband code-division mutliple access (WCDMA) Universal Mobile Telecommunications System (UMTS) and 1xEV-DO (cdma2000) wireless access technologies. The radio channels studied (by way of channel simulator) were static, typical urban environment with 50 km/h mobility (TU50) and a rural area environment with 100 km/h mobility (RA100). Carrier-to-interference ratios (CIRs) were varied.

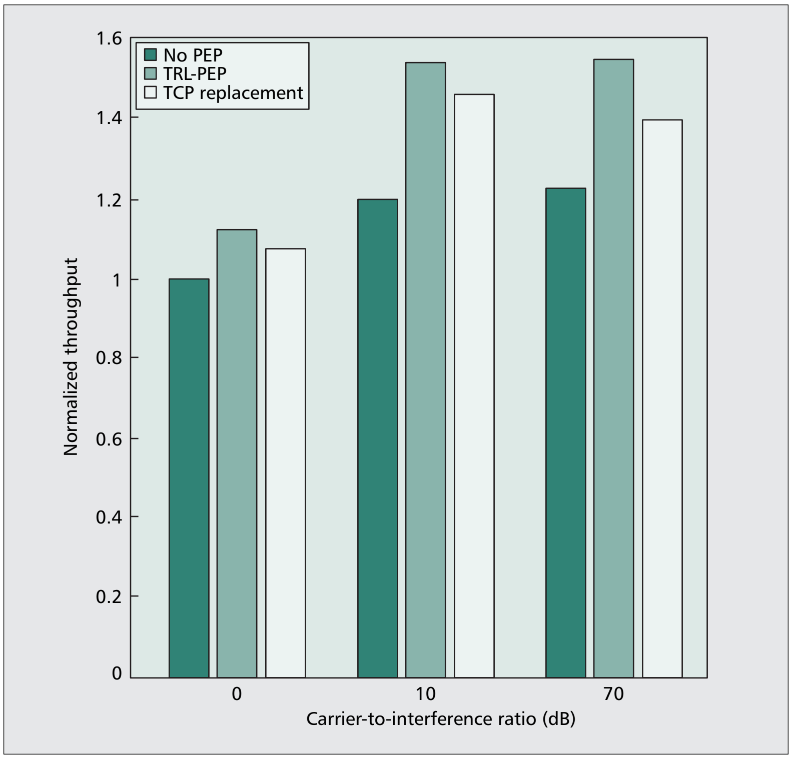
FTP throughput tests were conducted using the standard Windows FTP client with default receiver Awin optimally set to the given radio technology's bandwidth delay product. A Linux FTP server was used to study small (50 kbytes), medium (1 Mbyte), and large (4 Mbytes) file transfers. Throughput measurements were computed using the tcptrace analysis package. The WAPT utility (v3.0) was used to automate Web page download tests. The same Web page was used throughout all tests, and consisted of text, a textured image background, two 10-kbyte images, two 20-kbyte images, three 40-kbyte images, and two 80-kbyte images. WAPT provided the functionality to repeatedly and independently download the test page and record each page transaction time. Each data point was constructed by averaging 40 runs. All image files used in the tests were already maximally compressed, while all larger test files were created to be uncompressible.

Tests were conducted comparing:
- The standard Linux BIC TCP implementation [15] (an improvement on standard TCP, including the use of SACK options)
- The TRL-PEP
- A commercially available product relying on the TCP replacement approach (without content compression)

Figure 5 shows the 1-Mbyte file size FTP results obtained using WCDMA as the cellular radio access technology with a static fading model. The TRL-PEP provides up to ~25 percent improvement in throughput over standard TCP and also slightly exceeds the performance of the TCP replacement product. The primary reason for the improvement is the TRL-PEP's quick utilization of the dynamically expanding available bandwidth of the WCDMA link; conversely, standard TCP takes a long time to do so via slow-start. The counterintuitive result of TCP



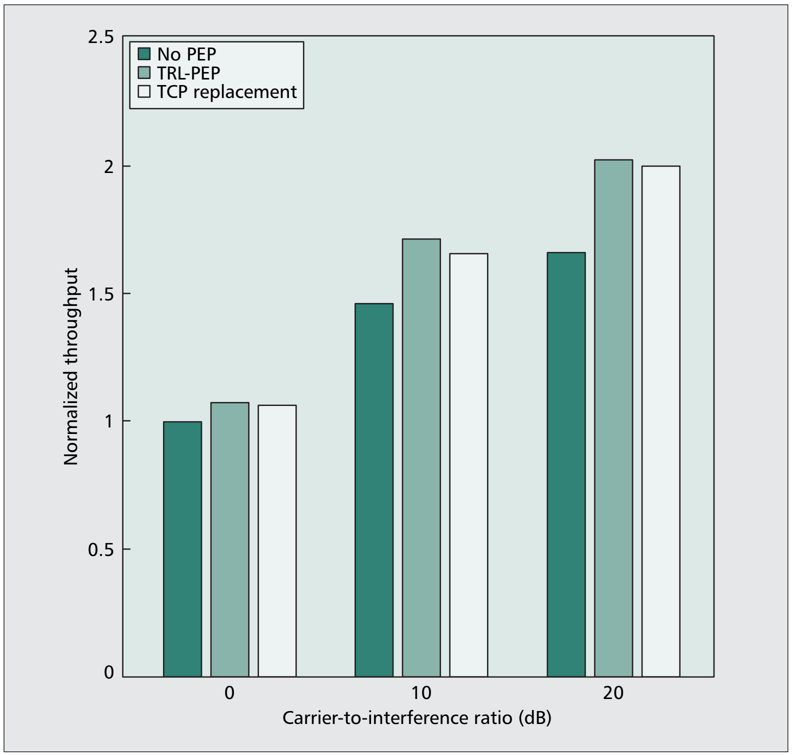
■ **Figure 4.** *1MB FTP download, 1xRTT.*



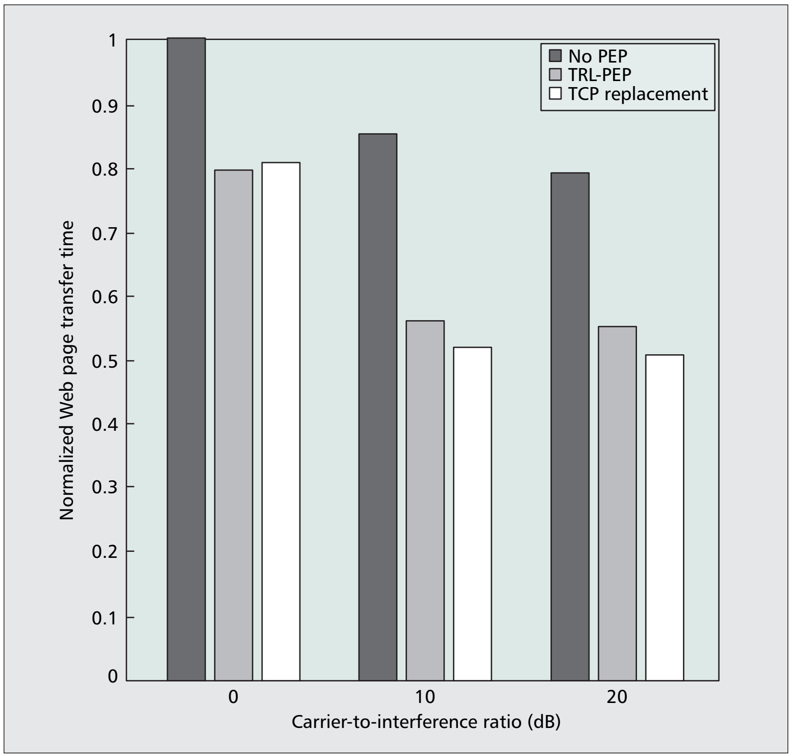
■ **Figure 5.** *WCDMA, static, 1-Mbyte FTP download.*

replacement product throughput at 70 dB CIR being marginally worse than at 10 dB CIR may be attributed to the variability of simulated radio channels. This discrepancy falls within the 95 percent confidence intervals computed for each data point (not shown).

Figure 6 shows the 4-Mbyte FTP results obtained using 1xEV-DO in an RA-100 channel. In this case the throughput improvement of TRL-PEP over standard TCP is up to ~20 percent, and it still marginally exceeds the performance of the TCP replacement product, again due to aggressive wireless link utilization and novel data recovery algorithms in the presence of packet losses. Figure 7 shows the results of the retrieval of the 300-kbyte test Web page in the static fading radio channel over 1xEV-DO. The results indicate up to a 35 percent reduction in the overall time required by the TRL-PEP to retrieve the Web page over standard TCP (as measured from the time the request is sent to the time the connection is completed). In this instance the performance of the TCP replacement product is marginally better than that of the TRL-PEP, but similar to Figs. 5 and 6, the two remain very close. In this Web-based test the improvement due to aggressive utilization of the link by the TRL-PEP is made more significant because multiple TCP connections are required to retrieve the Web page data. With each of these connections able to bypass the throughput-wasteful slow-start phase, the TRL-PEP is able to achieve the reported gains.



■ **Figure 6.** *1xEV-DO, RA-100, 4-Mbyte FTP download.*



■ **Figure 7.** *1xEV-DO, static, 300-kbyte Web download.*

CONCLUSION

We have surveyed a selection of different approaches to managing TCP performance over wireless links, and demonstrated that using freely available knowledge and tools, it is possible to construct a TCP performance enhancing proxy using semantics already built into the TCP standard. Our proposed solution has been shown to be fully customizable to suit different wireless access technologies and achieve excellent throughput improvements over standard TCP while maintaining transparency to both TCP endpoints. A physical prototype has been built and via benchmarking shown to outperform both standard TCP and, in the majority of test cases, a commercially available product based on the arguably more complex TCP replacement approach. The benchmarking was performed in the setting of a live network trial using two popular cellular radio access technologies based on the cdma2000 and UMTS third-generation standards.

ACKNOWLEDGMENTS

Jonathan Li was the primary contributor to this work during his time at Telstra. We acknowledge the tireless work of John Sekfy, technical staff at Telstra's Parramatta exchange, for helping to set up the experimental testbed, and Michael Hesse for his troubleshooting skills and device procurement assistance.

REFERENCES

[1] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Commun. Mag.*, vol. 41, no. 10, Oct. 2003, pp. 74–80.

[2] V. Srivastava and M. Motani, "Cross-Layer Design: A Survey and the Road Ahead," *IEEE Commun. Mag.*, vol. 43, 2005, pp. 112–19.

[3] S. Ladha *et al.*, "On the Prevalence and Evaluation of Recent TCP Enhancements," *Proc. IEEE GLOBECOM '04*, vol. 3, Dec. 2004, pp. 1301–07.

[4] M. Mathis *et al.*, "TCP Selective Acknowledgment Options," RFC 2018. 1996, Network Working Group.

[5] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on A Global Internet," *IEEE JSAC*, vol. 13, 1995, pp. 1465–80.
[6] C. Jin, S. Low, and X. Wei, "Method and Apparatus for Network Congestion Control," U.S. Patent & Trademark Office, Jan. 27, 2005.
[7] M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 26, Oc. 1996, pp. 281–91.
[8] M. Gerla *et al.*, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation," *Proc. IEEE GLOBECOM '01*, vol. 3, Dec. 2001, pp. 1698–1702.
[9] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for Wireless IP Communications," *IEEE JSAC*, vol. 22, no. 4, May 2004, pp. 747–56.
[10] S. Vangala and M. A. Labrador, "The TCP SACK-Aware Snoop Protocol for TCP over Wireless Networks," *Proc. VTC '03*, 2003, pp. 2624–28.
[11] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM Comp. Commun. Rev.*, vol. 27, no. 5, 1997, pp. 19–43.
[12] Fourelle Systems, Inc., "The Intelligent Solution to Optimizing Internet Bandwidth," Tech. White Paper, http://www.techmarketingink.com/Fourelle_wp.PDF, 2000.
[13] O. Shaham *et al.*, "Rate Control for Advanced Wireless Networks," *Proc. Int'l. Conf. 3rd Generation Wireless and Beyond*, June 2001, pp. 444–49.
[14] F. Khafizov and M. Yavuz, "Running TCP over IS-2000," *Proc. IEEE ICC '02*, vol. 5, May 2002, pp. 3444–48.
[15] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks," *Proc. INFOCOM '04*, vol. 4, Mar. 2004, pp. 2514–24.

BIOGRAPHIES

MILOSH IVANOVICH [SM'06] fills the role of senior emerging technology specialist within the Chief Technology Office of Telstra, and is an honorary research fellow at Melbourne and Monash Universities, Australia. His interests lie in queuing theory, teletraffic modeling, performance analysis of wireless networks, and the study and enhancement of TCP/IP in hybrid fixed/wireless environments. He obtained B.E. (1st class honors) in electrical and computer systems engineering (1995), a Master of Computing (1996), and a Ph.D. in information technology (1998), all from Monash University. He is an author of an edited book chapter, a patent, and 40 international journal and conference publications.

PHILIP W. BICKERDIKE [M] has held the position of emerging technology specialist within the Telstra Chief Technology Office since 2006. Prior to this, he was a research specialist at Telstra Research Labs — a role which included analysis and investigation into performance optimization for wireless data networks, wireless network dimensioning, development of network simulations, and study of performance enhancing proxies. He completed a B.E. (1st class honors) in computronics and a B.Comp. in computer science/software development at Deakin University in 2002.

JONATHAN C. LI [S'07] (jcli@ee.unimelb.edu.au) has been a Ph.D. student since 2006 working with the National ICT Australia (NICTA) Victoria Research Laboratory on the Managing and Monitoring the Internet (MAMI) project. His research is currently focused on cross-layer design in all-optical networks. Prior to this, he worked for Telstra Research Laboratories from 2002 in radio network performance optimization, conducting simulations into radio access MAC layer and TCP enhancement over wireless networks (an area in which he held a patent), as well as user perceived quality of service evaluation. He received a B.E. (1st class honors) in electrical and electronic engineering in 2001, and a B.Sc. in computer science and information technology systems in 1999 from the University of Western Australia.