Cross-layer enhancement of TCP split-connections over satellites links*

Fei Peng‡, Lijuan Wu§ and Victor C. M. Leung*†

Department of Electrical and Computer Engineering, The University of British Columbia, 2332 Main Mall, Vancouver, BC, Canada V6T 1Z4

SUMMARY

Satellites have played an important role in global telecommunications. However, transmission control protocol (TCP) suffers performance degradations over satellite links due to the long propagation delay and high bit-error rate. Among methods proposed to alleviate the impact of satellite link characteristics on TCP performance, split TCP connections separated by performance enhancement proxies (PEPs) have been proven attractive because they can keep TCP configurations in the end systems unchanged. While many protocols applied over satellite links between PEPs can effectively mitigate the effect of high loss rates, the long latency still limits the effectiveness of congestion control in these protocols. To minimize the effect of congestion we propose to take advantage of active queue management at the medium access control layer to provide immediate cross-layer feedback to the TCP segment over satellite. This approach also achieves proper differentiation between packet losses due to channel errors and congestion. Simulation results show that our novel mechanism can give substantial improvements in TCP performance over satellite networks, compared to the best performing version of TCP with or without PEPs. Copyright © 2006 John Wiley & Sons, Ltd.

Received 28 January 2006; Revised 1 May 2006; Accepted 5 July 2006

KEY WORDS: satellite networks; TCP; performance enhancement proxies; cross-layer enhancement

‡Correspondence to: Victor C. M. Leung, Department of Electrical and Computer Engineering, The University of British Columbia, 2332 Main Mall, Vancouver, BC, Canada V6T 1Z4.
§E-mail: feip@ece.ubc.ca
†E-mail: lijuanw@ece.ubc.ca
*Contract/grant sponsor: Canadian Institute for Telecommunications Research
Contract/grant sponsor: Canadian NSERC/CSA; contract/grant number: CSA223232-98

Copyright © 2006 John Wiley & Sons, Ltd.
1. INTRODUCTION

Owing to their large coverage areas, geosynchronous earth orbit (GEO) satellites can provide efficient data delivery to large numbers of geographically dispersed subscribers and hence provide an attractive means to offer high-speed Internet access to remote users. GEO satellite channels are characterized by long propagation delays (approximately 0.25 s), large bandwidth-delay products, and bit error rates (BERs) higher than fibre optic channels. Contemporary satellite links have typical BERs on the order of $10^{-7}$ or less [1]. The long propagation delay also result in a large bandwidth delay product, and hence the need to keep a large number of packets ‘in flight’ to fully utilize a satellite link.

Most Internet applications utilize the transmission control protocol (TCP) for reliable end-to-end data transport. While TCP performs well over terrestrial networks, the inherent characteristics of satellite links often result in significant degradations of TCP performance. One problem is that the long round trip time (RTT) causes TCP connections that include satellite segments to take up to several seconds before reacting to or recovering from congestions [1, 2]. Another problem is packet losses over satellite links. In terrestrial networks where packet losses are mainly caused by congestion, TCP responds by reducing its congestion window accordingly. This can cause unnecessary throughput reduction over a satellite link where packet losses may also be caused by transmission errors. However, congestion control is still necessary for reliable data transport over satellite networks. Therefore, proper differentiation between congestion losses and transmission losses would be necessary. Since TCP has been widely adopted by Internet hosts, many solutions have been proposed to overcome the problems of TCP over satellites stated above. The possible solutions can be classified into three main categories: link layer solutions, end-to-end solutions, and performance enhancement proxy (PEP) solutions [3].

Link layer solutions include the use of automatic repeat request (ARQ) and forward error correction (FEC) to mitigate packet losses due to transmission errors over the satellite link. However, these methods cannot solve the problems experienced by TCP that are caused by the long RTT over satellite links. In fact, using link-layer ARQ will cause large variations of the already very large RTT experienced by TCP connections over satellite links. Some end-to-end protocols have been specially designed for the satellite environment, such as TCP-Peach + [4], TCP Westwood with Bulk Repeat (TCPW BR) [5] and space communications protocol specification-transport protocol (SCPS-TP) [6]. Other end-to-end solutions [2] have been developed as extensions to the current TCP version [7]. The effectiveness of these solutions is limited by the fact that they are not universally supported by all end systems over the Internet. While it is reasonable to deploy these improved end-to-end protocols or TCP extensions in user terminals accessing the Internet via satellites, it may not be practicable to deploy these protocols in all end systems (e.g. web servers) that these user terminals may communicate with over the Internet. Instead, standard TCP [7] is globally deployed in all end systems. The PEP approach is attracting much attention nowadays as an effective method to improve TCP performance in satellite networks, as PEPs can be configured to act on behalf of the end systems without changing their TCP configurations. Two kinds of TCP PEPs have been proposed: TCP spoofing proxies [8] and TCP split connection proxies [3]. By locally acknowledging TCP segments, spoofing proxies solve the problem of slow startup speed of TCP over networks with long delays. However, the problems of long congestion control feedback delay and inability to differentiate between packet losses due to congestion or transmission errors remain. In contrast, TCP split connection proxies can employ a customized protocol over the satellite link to
compensate for specific link characteristics that would otherwise cause poor performance. Link layer solutions and end-to-end solutions can be combined with split connection proxies.

A pair of TCP split connection PEPs are usually employed to partition an end-to-end TCP connection into cascading satellite and terrestrial segments. The terrestrial segment conforms to the standard TCP protocol [7] to guarantee compatibility with all Internet hosts or end systems, while the protocol for the satellite segment can be tailored to the characteristics of the satellite link. A number of such protocols have been proposed in earlier work. For private satellite networks, the use of User Datagram Protocol has been proposed [9] as a means to avoid TCP flow control. However, it is limited to using time-outs for error recovery, and it is unable to cope with situations where congestion does occur over shared satellite links. In these situations, flow control is still required for reliable data transport. Other proposals [10, 11] use TCP Reno combined with TCP extensions to enhance the performance of TCP. However, most of these TCP extensions are unable to distinguish between packet losses due to congestion and transmission errors. Another well-known proposal for TCP split-connections uses the satellite transport protocol (STP) [12], which is optimized for high latency, asymmetric links with high packet loss rates. STP uses negative acknowledgements to speed up packet loss recovery, and it can correctly differentiate between congestion and packet losses due to transmission errors. The main drawbacks of STP are that it requires a customized implementation that is incompatible with TCP implementations. PETRA [13] enhances STP with a proxy congestion control protocol. It introduces a new layer over STP to preserve end-to-end semantics. Most of these PEP protocols are effective in overcoming problems stemming from a large bandwidth delay product and improving slow start performance. However, due to a very high RTT over satellite, and the feedback congestion control taking one RTT after congestion is detected before it becomes effective at the point of congestion, the buffer size employed at a split connection PEP has to be as large as the order of the bandwidth delay products [14]. PEPs can also be deployed onboard satellites [15] to further subdivide the connection into earth-to-space and space-to-earth connections and further reduce the RTT over each segment by approximately one half, thus reducing the buffer size over each segment. However, this alternative requires satellites with advanced onboard processing capabilities, which are not commonly available.

The above discussions motivate us to develop a cross-layer congestion control mechanism for the proxy service over a satellite network employing TCP split-connections, which can be implemented as a simple extension of standard TCP implementations. To control congestion effectively, we use random early detection (RED) active queue management [16] at the medium access control (MAC) layer to provide a cross-layer congestion indication to the TCP virtual source at the PEP. This immediate congestion feedback mechanism virtually eliminates congestion control delay and separates the TCP flow control and error recovery mechanisms. The proposed method is possible due to the collocation of the uplink bottleneck (the MAC layer buffer) and the TCP virtual source at the satellite gateways incorporating the PEPs. As far as we know, the proposed approach is original in the context of satellite networking, although cross-layer design involving TCP and the MAC layer in terrestrial wireless networks has been widely studied [17, 18]. Following the TCP selective acknowledgement (SACK) standard [19], we adopt the SACK option field in our mechanism to improve loss recovery capabilities, and refer to the proposed method as cross-layer SACK (CSACK). We also adopt the window scaling option [20] to overcome the window size limitation and improve throughput over the satellite link. The proposed method is applied in the system architecture described in the next section. We evaluate the performance of the proposed mechanism by simulations and show that it gives substantial
performance improvements over end-to-end and split TCP connections employing TCP SACK when the satellite link is utilized by a single TCP connection. Smaller but still significant performance improvements are obtained when multiple TCP connections share a common satellite link.

2. SYSTEM ARCHITECTURE AND PROPOSED MECHANISM

2.1. Split-connection architecture

Figure 1 illustrates the system architecture for the deployment of the proposed CSACK method. The satellite network provides an intermediate link in the end-to-end connection. The MAC protocol employed is multi-frequency time division multiple access, which is commonly employed over satellite networks. We consider a simple bent-pipe satellite that relays packets received from the uplink to the downlink without demodulation or error checking. There are two gateway satellite earth stations at either ends of the satellite link. The TCP split-connection is achieved by incorporating a split connection proxy in each gateway that serves as a PEP for the respective end host, thus isolating the host from the characteristics of the satellite link. When one of the end hosts requests to set up a TCP connection, its PEP intercepts this request and sets up a cascading TCP connection for it over the satellite link to the opposite PEP, which in turn sets up another cascading TCP connection to the corresponding end host. The PEPs are responsible for local acknowledgements and local retransmissions on behalf of end hosts over each of the terrestrial and satellite segments.

2.2. Random early detection queue

One of the novel features of our approach is that we deploy a RED queue at the uplink MAC layer in the gateway satellite earth station. The RED queue algorithm can detect incipient

![Figure 1. TCP split connection system architecture.](image-url)
congestion over the satellite network and manage the queue in a more active manner. In the network architecture considered, congestion tends to occur at the shared uplink due to the bent-pipe transponder and the fact that the terrestrial links at either end generally have much higher bandwidth (e.g. 100 Mbps or 1 Gbps Ethernet) than the satellite link. One of the main goal of the RED queue is to simultaneously achieve high throughput and low average delay [16].

A RED queue is characterized by a set of parameters, the most important ones being the two thresholds: minthresh and maxthresh, used to switch the queue management algorithm between different stages. A RED queue also maintains an estimate of the average queue length, $Q_{avg}$, which is calculated using an exponentially weighted moving average with weight $w$ as $Q_{avg} = (1-q_w) \times Q_{avg} + q_w \times q$, where $q$ is current queue length. When minthresh $\leq Q_{avg} \leq$ maxthresh, incoming packets are randomly marked, with a probability $p_a$ that is expressed as $p_a = (1 - count \times p_b)$ where $p_b = \max_p \times (Q_{avg} - \text{minthresh}) / \text{maxthresh} - \text{minthresh}$, $\max_p$ is the maximum value for $p_b$, and count is the number of packets since the last marked packet (count is reset to 0 each time a packet is marked and increases afterwards). Unlike Reference [16], we do not adapt $\max_p$ to the past history of average queue size. Incoming packets are marked with probability of 1 when $Q_{avg} > \text{maxthresh}$.

Normally the RED queue must be used together with TCP nodes supporting explicit congestion notification (ECN) [21]. If TCP gets an ECN-marked acknowledgement (ACK), TCP realizes that the network is congested and cuts its window by one half. However, if TCP is not in fast recovery/transmit phase or ECN action phase, it will cut its congestion window when it encounters a packet loss, even if the packet loss is caused by transmission errors. In this case, TCP cannot differentiate whether this packet loss is due to error or congestion. Another shortcoming of ECN is that it takes an RTT for the sender’s reaction to become effective at the congested node as the ECN needs to be turned around by the TCP receiver. The effectiveness of ECN over satellite networks is therefore limited due to the very long RTT, which can be more than 500 ms in GEO satellite systems, although ECN can be effectively applied over terrestrial wireless networks [21].

As the satellite uplink is usually the bandwidth bottleneck, it seems obvious to use RED queue at the MAC layer of the satellite gateway. However, to our best knowledge this approach has not been considered in the satellite networking literature. In our proposed mechanism, the RED algorithm is used to actively manage the MAC queue and provide immediate cross-layer feedback to the TCP virtual sources at the PEP, to improve throughput while reducing delay.

2.3. Proposed cross-layer SACK mechanism for TCP

In the split-connection scenario as shown in Figure 1, the PEPs act as both a virtual receiver and a virtual source of the cascading TCP connections, and the satellite link functions as a pipe with no further queuing delay in between. So the PEPs have access to local information from the collocated gateways concerning underlying queue conditions. Due to the scarcity of satellite bandwidth shared by multiple subscribers and multiple TCP connections and the availability of high bandwidth terrestrial links at reasonable cost, the traffic bottleneck between the PEPs is likely to occur at the MAC layer of the satellite uplink. By exchanging information between the TCP and MAC layers, the TCP virtual sources at the PEP can react to the RED information immediately. Therefore, the PEP can control congestion and prevent queue overflow much more efficiently without the long RTT delay.
We adopt the RED algorithm as discussed above for congestion detection at the MAC queue, but take different actions according to the congestion information. Based on comparisons between $Q_{\text{avg}}$, minthresh and maxthresh at the MAC queue, instead of marking/dropping packets, the MAC layer generates congestion notification signals for the TCP virtual source at the PEP, as shown in Figure 2. The congestion signal can be implemented by a flagged interrupt in the gateway's real-time control software. For congestion control, TCP can react to this notification instead of packet losses, and reduce the congestion window accordingly. Specifically, when a packet arrives at the MAC queue, depending on the value of $Q_{\text{avg}}$, congestion signal I (analogous to marking) is sent to TCP with a probability that increases linearly from 0 to 10% for $\text{minthresh} \leq Q_{\text{avg}} \leq \text{maxthresh}$, or congestion signal II (analogous to dropping) is sent to TCP with a probability that increases linearly from 10 to 100% for $\text{maxthresh} \leq Q_{\text{avg}} \leq 2 \times \text{maxthresh}$. No congestion signal is sent if $Q_{\text{avg}} \leq \text{minthresh}$, and congestion signal II is always sent if $Q_{\text{avg}} > 2 \times \text{maxthresh}$. These congestion notification signals enable immediate feedback of congestion information. In the above cases, packets are dropped when the MAC queue overflows.

When a TCP virtual source at the PEP receives an ACK, no matter whether it is a new ACK or a duplicate ACK, it will first look for the congestion notification signal. In the presence of congestion signal I or II, the source cuts its congestion window by $\frac{1}{2}$ or $\frac{1}{4}$, respectively, and clears the congestion signal. Otherwise, the source allows its window to grow as in the original TCP mechanism. Compared to the long delay of feedback congestion control employing ECN or otherwise, the immediate cross-layer feedback of congestion information can significantly speed up control action at the TCP virtual source and improve performance. By using congestion signals to control the reduction of the congestion window, congestion control and error recovery functions are separated at the TCP virtual source. As usual, the TCP virtual source recovers lost segments using either fast retransmission after receiving three duplicate ACKs, or retransmitting a lost segment after a retransmission timeout (RTO).

In a satellite network, when the link suffers from a high error rate and multiple segment losses occur within one window of data, the throughput degradation is still large. TCP SACK option [19] is effective for quick recovery of transmission errors [23] and is recommended for use over satellite links. In this paper, we consider that TCP SACK is used over the satellite network between the PEPs. We also adopt the window scale option [20] to enable a sufficiently large window size required by the large bandwidth delay product over the satellite link. We call the proposed mechanism CSACK, which combines cross-layer congestion control feedback with TCP SACK and window scale options. To ensure global compatibility with existing TCP versions, CSACK is implemented in PEP nodes only, while end nodes are not modified.
3. SIMULATION RESULTS

We use the network simulator, ns2 [24], as the simulation tool. The system configurations evaluated include a single TCP connection and multiple TCP connections sharing a duplex satellite link. The TCP throughput is defined as the number of data bits (not including TCP/IP headers) received divided by the time used to finish the transmission. The simulation parameters are listed in Table I. The parameters chosen for the RED algorithm is based on Reference [25]. We obtain simulation results for different protocol configurations under the same network and traffic conditions to compare the performance of:

1. ‘Split CSACK’, the proposed CSACK mechanism operating in TCP split connections between PEPs.
2. ‘End to end SACK’, where both end hosts support the TCP SACK option and PEPs are not used.
3. Split SACK, where the standard SACK option is used between the PEPs only.
4. ‘Split Immediate Feedback’ where the TCP virtual source at the PEP only checks the congestion notification signal when it receives a new ACK, and it cuts the congestion window according to both packet losses and the local congestion notification signals.
5. ‘Split Error Identification’, where no immediate feedback is employed, and TCP uses ECN with the RED mechanism to provide feedback after an RTT over the satellite network and cut the congestion window by one half when ECN is set.

The last two configurations are intended to test individually the effects of the two mechanisms, i.e. immediate feedback and error identification, that are incorporated in the proposed CSACK, and will be explained in more details later.

3.1. Single connection case

First, we compare the performance of the proposed mechanism with TCP SACK applied end-to-end and between PEPs in a split connection, with and without bit errors in the satellite link. We further demonstrate the advantages of our proposed CSACK mechanism by evaluating the average RED queue length and the individual effects of the two schemes incorporated in CSACK: immediate feedback and error identification.

<table>
<thead>
<tr>
<th>Parameter item</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth for terrestrial link</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>Propagation delay for terrestrial link</td>
<td>50 ms</td>
</tr>
<tr>
<td>Bandwidth for gateway over satellite</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>Propagation delay for satellite link</td>
<td>250 ms</td>
</tr>
<tr>
<td>Packet length for downstream (including TCP/IP headers)</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>Offered window size for terrestrial and satellite links</td>
<td>32768 kB</td>
</tr>
<tr>
<td>RED queue buffer capacity (monitor in packets)</td>
<td>125 packets</td>
</tr>
<tr>
<td>minthresh for RED Queue</td>
<td>5 packets</td>
</tr>
<tr>
<td>maxthresh for RED Queue</td>
<td>15 packets</td>
</tr>
<tr>
<td>maxp for RED Queue</td>
<td>0.1</td>
</tr>
<tr>
<td>q_w for RED Queue</td>
<td>0.002</td>
</tr>
</tbody>
</table>
Figure 3 shows the throughput of a single TCP connection versus the file size without bit errors. The file transfer protocol (FTP) is used as the application data source. All three curves show that the throughput increases with the file size because the impact of the slow-start is decreased. If the transmission is long, the slow-start only takes up a small part of the overall transmission time and most of the data is transmitted in the congestion avoidance phase, thus efficiently utilizing the available bandwidth. For end-to-end SACK, the throughput performance is lower than Split SACK when the file size is small because of the slow-start effect. However, when the file size becomes larger, the detrimental effect of slow-start is reduced and the throughput of end-to-end SACK is getting closer to that of Split SACK. Overall, the proposed Split CSACK mechanism outperforms the Split SACK configuration when the file size is larger than 100 kB; for smaller files sizes, the queue occupancy is too small for the congestion control mechanism to take effect. For example, when file size is 10 MB, the throughput of Split CSACK is about 35% higher than that of Split SACK.

We investigate the impact of different BERs over the satellite link on TCP performance. The nominal BER of a contemporary satellite link is of the order $10^{-7}$–$10^{-9}$, but could degrade to $10^{-6}$ or higher during rain fades. The file size used in the simulations is 10 MB. To evaluate individual impact, we run simulations for the immediate feedback mechanism and error identification mechanism separately. With immediate feedback, the congestion signal is generated as we discussed before, but TCP only checks the congestion signal when it receives a new ACK, and cuts the congestion window according to average queue length. If TCP receives
a duplicate ACK, it enters fast transmit and fast recovery as usual and ignore the congestion signal. We denote this scheme as the split immediate feedback in the graph. For error identification, we consider that nodes are ECN-capable and employ RED queues. The ECN marks are returned by the TCP receiver to the TCP sender after an RTT over the satellite network. To differentiate the source of packet loss, we employ the method in Reference [26]. ECN is added to duplicate ACKs that is caused by congestion. TCP cuts the congestion window by one half, if one or more of duplicate ACKs contain an *echo-flag*. Otherwise, if an ACK containing an *echo-flag* is not duplicated, the congestion window is kept unchanged.

From Figure 4, we can see that the throughput performance of Split SACK is better than end-to-end SACK thanks to local retransmissions for error recovery. Even though Split SACK can recover several packet losses within an RTT, the performance degrades sharply when the BER reaches $10^{-6}$. On the other hand, the negative impact of increased BER on CSACK throughput is much smaller because it can check the status of the underlying queue to distinguish between congestion loss and transmission loss. When the BER is $10^{-6}$, the throughput of split CSACK is five times higher than that of split SACK and also better than that of end-to-end SACK.

We can see the effect of different mechanisms more clearly from the split immediate feedback and split error identification curves. Both of these mechanisms can improve TCP performance to some degree; however, their respective performance stays lower than the proposed CSACK, which combines these two mechanisms. For example, the throughput of split immediate feedback degrades sharply when the BER is higher than $10^{-8}$. When the BER is $10^{-6}$, split error identification yields almost the same throughput as Split CSACK, while immediate feedback...
performs much more poorly. This is because split immediate feedback cannot distinguish between transmission errors and congestion losses. Under this condition \((BER = 10^{-6})\), Figure 5 shows that the average queue length exceeds \text{minthresh} (5 packets) only once, which means that immediate feedback takes effect only once during the simulation. The lower buffer occupancy is caused by TCP cutting its congestion window due to transmission losses. However, when the BER is \(10^{-8}\) or lower, split immediate feedback can raise the throughput by about 34% compared to split SACK, by taking timely actions to prevent possible network congestion. The overall performance of split CSACK results from the combined effects of these two mechanisms, thus yielding superior performance regardless of whether there is network congestion or high BER.

3.2. Multiple connections case

Figure 6 illustrates the total TCP throughput over an error-free channel versus the number of TCP connections, which varies from 10 to 40. Each connection simultaneously transmits a 200 kB file, which is larger than the size of most HTTP objects. This allows the general trend that the overall throughput improves with an increasing number of TCP connections to be illustrated with a reasonably small number of TCP connections. With a smaller file size that is more representative of HTTP objects, a larger number of TCP connections would need to be simulated. As the satellite link is error-free in this scenario, packet losses are caused by congestion only. Note that owing to the immediate feedback, the packet losses in Split CSACK due to congestion are much less than that of split SACK. Therefore, the performance of CSACK is better than both end-to-end SACK and split SACK.

We also conduct simulations for multiple TCP connection cases over a range of the BER from \(10^{-6}\) to \(10^{-9}\). The results for 20 TCP connections simultaneously transmitting 200 kB files are
shown in Figure 7. Evidently, the proposed CSACK method yields the highest throughput compared to the other four methods over the entire range of BER values. The performance gains over split SACK and end-to-end SACK are especially large when the BER becomes large. However, the percentage improvements of CSACK over these two schemes for multiple TCP connections are not as high as those for a single TCP connection. Multiple connections are usually more effective in utilizing the total bandwidth available, because a bandwidth reduction caused by the reduced transmit window of one connection (e.g. due to bit-errors) is matched by increased transmit windows and hence bandwidth expansions by other connections. Thus the detrimental effects of BER on end-to-end SACK and split SACK is less pronounced in the multiple TCP connection cases compared to the single connection cases. Nevertheless, our simulation results show that CSACK still gives a more than 30% improvement in throughput compared to end-to-end SACK in multiple connections cases. Split SACK obtains better performance than end-to-end SACK where there are transmission losses since the reduced RTTs of split TCP connections help to alleviate the slow-start effect over long-delay satellite links.

As shown in Figure 7, split immediate feedback gives the same throughput performance as CSACK when the BER is smaller than $10^{-7}$, but its throughput is much reduced at higher BERs. On the other hand, it can be seen that split error identification is able to maintain its throughput performance over the full range of BER values; thus it can help to improve the overall performance when BER is high. Figure 8 shows that the average buffer occupancy of Split Immediate Feedback can exceed minthresh (five packets) many times over the 20 s
Figure 7. Impact of bit error rate on the total throughput of 20 TCP connections.

Figure 8. Queue occupancy (20 Flows, BER = 10^{-6}).
simulation period, so that the immediate feedback congestion control method can take effect more often compared with the single connection case.

4. CONCLUSIONS

To maintain compatibility with end systems widely deployed over the global Internet, the use of proxies to enhance TCP performance over satellite networks will remain a necessity until improved transport protocols are globally adopted. In this paper, we have proposed a novel cross-layer congestion control method, called CSACK, for TCP SACK split-connections applied to a satellite link between two performance enhancement proxies. The key feature of CSACK is local congestion notification from the MAC layer to the TCP layer in the PEP based on the current MAC buffer occupancy managed by the RED algorithm. The cross-layer congestion feedback is immediate and eliminates the long congestion feedback delay over satellite. Since the scheme is used at PEPs over split TCP connections, it has the advantage of requiring no modification to end systems and therefore is very practical for current applications. Simulation results show that the proposed CSACK mechanism offers a 600% improvement in the throughput of a single TCP connection over the satellite link with $10^{-6}$ BER, compared to end-to-end SACK. With 20 TCP connections sharing the satellite link and under the same BER, the throughput improvement is still better than 30%.

ACKNOWLEDGEMENTS

This work was supported by grants from Com Dev International and the Canadian Institute for Telecommunications Research under the NCE program of the Canadian Government, and by the Canadian NSERC/CSA Partnership Program under grant CSA223232-98.

REFERENCES


**AUTHORS’ BIOGRAPHIES**

**Fei Peng** received the MS degree and PhD degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1977 and 2000. From 2001 to 2003, she was a Postdoctoral fellow in the Department of Electrical & Computer Engineering at the University of British Columbia (UBC), where she is currently a Research Associate. Her current research interests are in Internet transport protocol and QoS performance over a wide diversity of integrated mobile, wireless and wired networks.

**Victor C. M. Leung** received the BASc (Hons.) and PhD degrees, both in electrical engineering, from the University of British Columbia (U.B.C.) in 1977 and 1981, respectively. He was the recipient of many academic awards, including the APEBC Gold Medal as the head of the 1977 graduate class in the Faculty of Applied Science, U.B.C., and the NSERC Postgraduate Scholarship. From 1981 to 1987, Dr Leung was a senior Member of Technical Staff and satellite systems specialist at MPR Teltech Ltd. In 1988, he was a Lecturer in Electronics at the Chinese University of Hong Kong. He returned to U.B.C. as a faculty member in 1989, where he is a Professor and holder of TELUS Mobility Research Chair in Advanced Telecommunications Engineering in the Department of Electrical and Computer Engineering. His research interests are in mobile systems and wireless networks. Dr Leung is a Fellow of IEEE and a voting member of ACM. He is an editor of the IEEE Transactions on Vehicular Technology, and an editor of the International Journal of Sensor Networks.