Enhancing TCP Splitting in Satellite-Terrestrial Networks via ACK Reservation

Jing Zhu[1], Sumit Roy[1], Jae Kim[2]
{zhuj, roy}@ee.washington.edu
jae.h.kim@pss.boeing.com
1. Dept. of Electrical Eng, Univ. of Washington, Box 352500 Seattle, WA 98195-2500
2. Phantom Works, The Boeing Co. P. O. Box 3999 MC 3W-51 Seattle, WA 98124

Abstract- In this paper, we focus on the performance of *TCP splitting* in satellite-terrestrial hybrid networks. By simulation we show that long-time link outage has great negative impact on TCP throughput. To solve the problem, an *ACK reservation* scheme is proposed to speed up TCP recovery from link outage and analytical estimate of a key parameter of the scheme is provided for achieving high end-to-end TCP throughput. Simulation results validate our analysis, and show that after the end of link failure, the TCP source can be re-started using ACK reservation with virtually no additional lag leading to great improvement on bandwidth utilization and end-to-end throughput.

Key Words: satellite networks, TCP/IP

I. Introduction

There exists considerable interest in ways to improve TCP performance over satellite networks, where both long propagation delay and packet losses in satellite channel impact TCP throughput negatively. One method is to use a performance enhancing proxy at the satellite channel access nodes that divides the end-to-end TCP connection between a (terrestrial) source and (airborne) destination pair into two or more segments; this approach is called *TCP Splitting* (I-TCP [1]). The Skyx protocol [1] is a commercial implementation of such a *TCP Splitting* approach.

Nevertheless, performance sensitivity issues arise with respect to the interaction among path segments and different layers. [3] studied the performance of TCP spoofing by simulation and showed the problem of data accumulating at the spoofer (i.e. access node), leading to a second bottleneck. Another problem is slow TCP recovery from long link outage events - if a satellite channel falls into a deep fade, all packets transmitted during the fade duration will be lost, implying an outage. But with *TCP Splitting*, TCP source is unaware of the event of link outage, and will continue sending packets till buffer overflow. This may lead to successive timeouts happen and consequent doubling of the RTO (Retransmission TimeOut) timer of TCP source can lead to a large value. As a result, the TCP source may still be "off" after the link recovers, significantly reducing bandwidth utilization.

In this paper, we focus on the problem of slow TCP recovery from long-time link outage. A solution called *ACK reservation* is proposed. Furthermore, analytical conditions are provided on a key protocol parameter for good TCP performance.

The paper is organized as follows. In Section 2, we introduce the system model of TCP splitting in satellite-terrestrial hybrid networks. Section 3 describes the impact of long TCP recovery time from link outage. Our proposal to enhance TCP splitting via ACK reservation is introduced in Section 4. Appropriate choice of key parameters of ACK reservation is analytically investigated in Section 5. Simulation results are reported in Section 6 to validate our analysis.

II. System Model

In Fig.1, the network scenario and system model for our study is shown. A TCP connection is divided into two parts by the satellite access node: a terrestrial and a satellite portion. In the terrestrial portion, a normal version of TCP (Reno) is used. In the satellite portion, improved link-layer protocol (ARQ, FEC, etc.) or some advanced version of TCP (SACK, FACK, etc.) can be used. In this paper, we assume a fully reliable selective repeat ARQ over the satellite link, where a data packet is not cleared from the send buffer until the arrival of

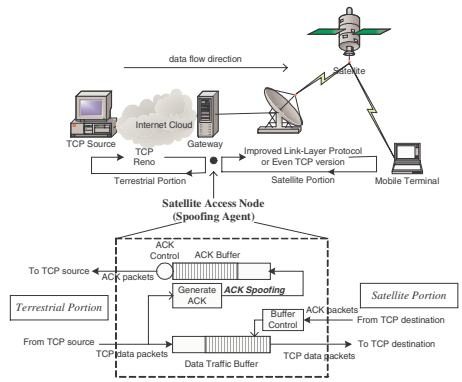

Fig. 1. Network scenario and system model for TCP splitting in a satellite network

‡: This work was supported in part by ARMY/CECOM under DAAB07-01-CL845 "OTM Wideband Satellite Communications Terminal with Techniques for Blockage Mitigation"

[1] http://www.mentat.com/skyx/skyx-gateway.html

its acknowledgment. Since the satellite portion is a point-to-point link, no flow control mechanism and congestion avoidance scheme is needed and the transmission rate is set equal to the link bandwidth.

The satellite access node plays a key role in determining end-to-end performance since it connects the two parts of the TCP session by acting as a spoofing agent for TCP source by generating ACK packets and forwarding data packets over the satellite link (including retransmission of lost packets) and clearing the buffer after receipt of acknowledgements from the destination. All enhancements with respect to TCP splitting are implemented at the satellite access node so that no modification is required to the TCP stack in the end systems.

III. Slow TCP Recovery from Long-Time Link Outage

Earlier work [2] indicates that satellite channels may experience persistent loss (outage) so that consecutive packets are corrupted over an extended period due to link blockage and/or interference. We include a simulation trace to highlight this issue. Please note that wireless random losses exist even without blockage. There are many link layer approaches in the literature to combat such errors e.g. FEC (Forward Error Correction) and ARQ (Auto-Repeat reQuest), which is not the focus of this paper. An error-free channel is assumed for the non-blockage duration.

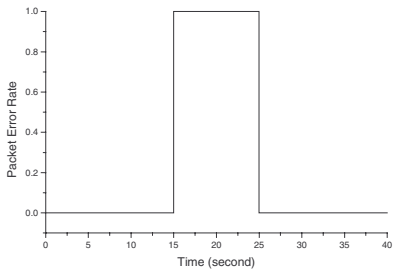

Fig. 2. Modeling the long time link outage of the satellite channel

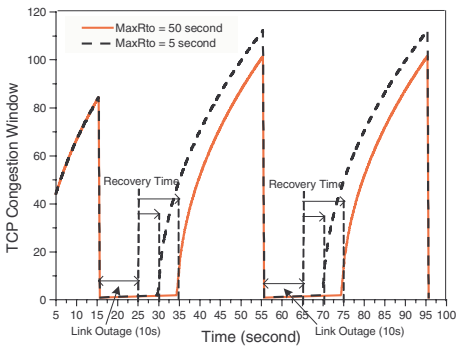

Fig. 3. Tracing TCP Congestion Window

In our simulation, the (one way) satellite propagation delay is 250 ms, the terrestrial propagation delay is 25ms, the data buffer size of the satellite access node is 150 (TCP packets), and a periodic process is used to model the link outage in the satellite channel where one cycle of the process is shown in Fig. 2. The period of link outage $T_o = 10s$, and the length of a cycle is $T_c = 40s$. Packet losses are only assumed to occur during a link outage, and no packets can be successfully transmitted in this period. Therefore, the maximum throughput in such channel is $1 - \frac{T_o}{T_c}$ (=0.75 for Fig.3).

From Fig. 3, we can see that although reducing maximum retransmission timeout time (MaxRto) from 50s to 5s does speed up TCP recovery, several seconds are still needed for the TCP source to "wake up" after the end of the link outage event.

IV. ACK Reservation Scheme

In this section, we introduce a solution to the problem of slow TCP recovery from long-time link outage. Without modifying TCP end systems, the only way to notify the TCP source about link recovery from outage is by using TCP ACKs. During the link outage, successive timeouts result in the threshold of TCP congestion avoidance decreasing to a very small value (say 1). Therefore, after detecting that the link has returned, TCP will restart from the congestion avoidance phase - thus every ACK increases the TCP congestion window by 1/cwnd, where cwnd is the current congestion window size. It follows that larger the number of ACK packets, the faster the recovery of the TCP congestion window. Thus we propose to *reserve* enough ACK packets at the beginning of a link outage event that are sent back to the source aggressively as soon as link recovery is detected, triggering the TCP source without further delay.

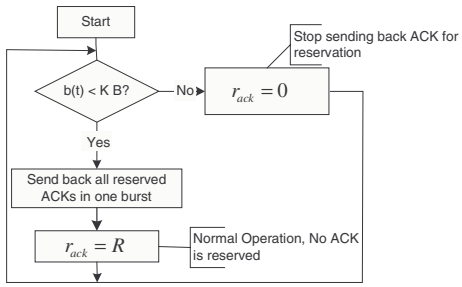

Fig. 4. Algorithm of ACK Reservation Scheme

We introduce some notation below for our subsequent analysis.

- r_{ack}: The output rate of TCP ACK packets;
- R: The rate of generating TCP ACK packets (this depends on the arrival rate of the TCP data packets);
- K: Threshold for ACK reservation $(0 < K < 1)$;
- B: The data buffer capacity of the satellite access node;
- $b(t)$: The queue length of the data buffer at time t $(0 \leq b(t) \leq B)$.

Flow chart Fig.4 illustrates our proposal. Generally, R is equal to the arrival rate of TCP data packets. If the buffer occupancy exceeds the parameter K $(0 < K < 1)$, ACK reservation is initiated and all ACK packets generated are reserved till the occupancy ratio falls below K. Obviously, the choice of K significantly impacts performance - too small value (unnecessarily) initiates ACK reservation too early and slows down

TCP window increase when the satellite channel is in good condition; too large value does not allow enough ACK packets to be reserved for fast recovery.

V. Performance Analysis of TCP Splitting with ACK Reservation

The parameters used in our following analysis are shown as follows. Note that some of the parameters appearing in the previous section will be still used in this section.

- μ: The satellite bandwidth.
- B_1: The maximum number of packets sent out from the satellite access node during a terrestrial round trip time ($B_1 = \mu T_1$).
- B_2: The bandwidth delay product of the satellite portion, ($B_2 = \mu T_2$).
- B_K: Unused buffer when ACK reservation starts ($B_K = B(1-K)$).
- $w(t)$: The congestion window size of the TCP source at time t.

A. Analysis of The Case without Link Outage

As is well known, the congestion window of the TCP source is always limited by the receiver window size, which equals the rest data buffer size at the access node. On the other hand, a packet in the buffer cannot be removed until the corresponding ACK is received. Therefore, to make full use of the satellite link, the number of packets in the buffer must be more than the bandwidth delay product of the link for all time yielding

$$b(t) \geq B_2. \tag{1}$$

With the ACK reservation scheme, as the queue length of the buffer reaches KB, all ACK packets generated subsequently will be reserved. When the buffer queue length falls below KB, the reserved ACK packets will be sent back to TCP source using the receiver window size value of B_K.

If $B_K < B_1$, the queue length in the buffer will be reduced by $B_1 - B_K$ after a terrestrial round trip time (RTT) and the remaining space in the buffer is then B_1. Subsequently, the queue length continues decreasing till the arrival of B_1 packets that occurs after another terrestrial RTT T_1. The buffer occupancy then decreases to a minimum possible value of $B - 2B_1$. To satisfy Eq.1, we must then have

$$B \geq 2B_1 + B_2. \tag{2}$$

If $B_K > B_1$, the queue length of the buffer will increase by $B_K - B_1$ after a terrestrial RTT. However, prior to the arrival of B_K packets, the queue length in the buffer continues decreasing to the minimum value $B - B_K - B_1$. In order to satisfy Eq.1, we must have

$$B \geq B_1 + B_2 + B_K. \tag{3}$$

Fig. 5 illustrates the above analysis, where cwnd denotes the congestion window of the TCP source.

Combining Eq.2 and 3, we have

$$B \geq B_1 + B_2 + \max(B_1, B_K). \tag{4}$$

In order to achieve the maximum throughput during the period without link failure, Eq.4 must be satisfied with the ACK reservation scheme.

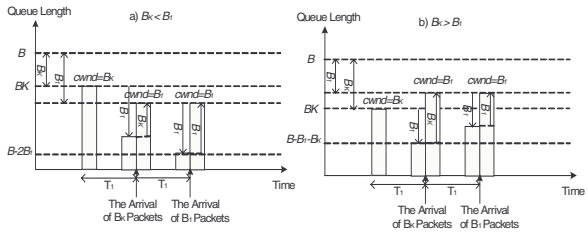

Fig. 5. Variation of The Queue Length in The Buffer

B. Analysis of The Case with Link Outage

Obviously, the threshold K dominates the speed of TCP recovery from link outage. The lower the threshold, the greater the number of reserved ACK packets and faster the speed of TCP recovery from outage. Nevertheless, if K is chosen too large, the performance will degrade because of too early initiation of ACK reservation, indicating that an optimal point exists.

After the link recovers, all reserved ACK packets are sent to TCP sender and TCP congestion window increases from 1 to some value (say w_0). Due to the long outage duration, successive timeouts reduce the TCP congestion avoidance threshold to a very small value (say 1) and the TCP source always stays in congestion avoidance phase. Thus we have

$$\frac{(w_0+1)}{2} w_0 = B_K. \tag{5}$$

If the congestion window of the TCP source after receipt of all reserved TCP ACKs is larger than μT_1, the required input rate of the buffer is seen to be satisfied and the recovery is complete. Otherwise, the time needed for recovery is the duration for the congestion window to be rebuilt up to μT_1. If the input rate of the date buffer increases to μ before the queue length falls below B_2, the buffer occupancy will be always larger than bandwidth delay product of the satellite link, so that full utilization of the satellite link can be achieved.

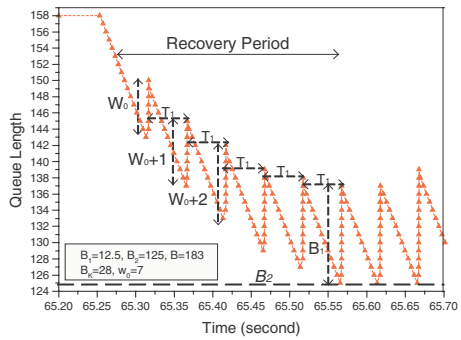

Fig. 6. Tracing the queue length of the buffer during the recovery period

In the following, we derive the condition for achieving maximum throughput, which implies that the queue length never falls below B_2. Fig. 6 shows a trace of the queue length of the buffer in the recovery phase - it is seen that the queue length just reaches B_2 at the end of the recovery. The time (denoted with T') required by the TCP source to increase the congestion window from w_0 to B_1 is given by

$$T' = (B_1 - w_0 + 1)T_1, \quad (w_0 < \mu T_1). \tag{6}$$

The number of packets from the TCP source, arriving at the buffer in duration T', is

$$w_0 + (w_0+1) + (w_0+2) + ... + (w_0+n-1) = nw_0 + \frac{n-1}{2}n, (n = \lceil \frac{T'}{T_1} \rceil).$$

Simultaneously, the number of packets removed from the buffer is $\mu T' + B_1$. Since the queue length never falls below B_2, we have

$$B - B_K + [nw_0 + \frac{n-1}{2}n] - (\mu T' + B_1) \geq B_2, \quad (n = \lceil \frac{T'}{T_1} \rceil). \quad (7)$$

Combining Eq. 5∼7, we get,

$$w_0^2 - B_1 w_0 - (B - B_2 - \frac{3B_1}{2} - \frac{B_1^2}{2}) \leq 0, \quad (8)$$

The sufficient condition of the above inequation having solution is

$$B \geq B_2 + \frac{(6B_1 + B_1^2)}{4}. \quad (9)$$

Combining Eq.9 with Eq.4 (the condition for acquiring the maximum throughput in the case without link outage must be satisfied as well), we have

$$B \geq B_2 + B_1 + \max(\frac{2B_1 + B_1^2}{4}, B_1, B_K). \quad (10)$$

In Eq.8, the left side of the inequation reaches the minimum with "$w_0 = \frac{B_1}{2}$". Therefore, although there may be a loose range of w_0 to fulfill Eq.8, it is for sure that "$w_0 = \frac{B_1}{2}$" must be one if the solution exists. Hence it is very safe to set B_K (using Eq.5) with "$w_0 = \frac{B_1}{2}$", leading to

$$B_K = \frac{(B_1+2)}{8}B_1. \quad (11)$$

Then Eq.10 turns into

$$\begin{aligned} B &\geq B_2 + B_1 + \max(\frac{2B_1 + B_1^2}{4}, B_1, \frac{B_1^2 + 2B_1}{8}) \\ &\geq B_2 + B_1 + \max(\frac{2B_1 + B_1^2}{4}, B_1). \end{aligned} \quad (12)$$

The sufficient condition of "$\frac{2B_1+B_1^2}{4} \geq B_1$" is "$B_1 \geq 2$". Since B_1 is the product of the satellite bandwidth and the terrestrial round trip time, "$B_1 \geq 2$" is true for most cases. Therefore Eq.12 turns into

$$B \geq B_2 + (\frac{6B_1 + B_1^2}{4}). \quad (13)$$

In conclusion, Eq.11 provides an optimal way to choose K. And the required buffering for achieving the full throughput in both cases is given by Eq.13. We define $B_2 + (\frac{6B_1+B_1^2}{4})$ as B_{min}, which is the minimum buffer size for the maximum performance. In practice, we set B with B_{min} $(= B_2 + \frac{6B_1+B_1^2}{4} = 183)$ in Fig.6 to meet the minimum requirement for full throughput.

On the other hand, if $Eq.13$ is not satisfied so that the queue length decreases to below B_1 during the recovery period, the maximum performance can not be acquired. The recovery time in this case is defined as the time for the queue length of the buffer to build up to the bandwidth delay product of the satellite link. We denote it with T_r. By using a reliable transmission protocol (TCP, ARQ, ...), the packets in the buffer can not be removed until the arrival of the acknowledgment. Therefore before the maximum throughput is reached, the output rate of the buffer at the satellite access node is determined by the input rate of the buffer, but with a round trip delay T_2. Denoting the output rate of the buffer at time t with $r_o(t)$ and the input rate with $r_i(t)$, we have

$$r_o(t) = r_i(t - T_2). \quad (14)$$

On the other hand, the input rate of the buffer is determined by the congestion window $w(t)$ of the TCP source. In congestion avoidance phase, $w(t)$ increases by one every round trip time, i.e.,

$$r_i(t) = r_i(t - T_2) + \frac{T_2}{T_1^2}. \quad (15)$$

Eq. 14 can thus be written

$$r_o(t) = r_i(t) - \frac{T_2}{T_1^2}; \quad (16)$$

At the epoch of the recovery end (denoted by t_e), the queue length of the buffer exceeds $B_2 (= \mu T_2)$ and the output rate of the buffer reaches its maximum μ. From Eq. 16, we have

$$r_o(t_e) = \mu, \quad (17)$$

$$\begin{aligned} r_i(t_e) &= r_o(t_e) + \frac{T_2}{T_1^2} = \mu + \frac{T_2}{T_1^2}, \\ w(t_e) &= (\mu + \frac{T_2}{T_1^2})T_1 = B_1 + \frac{T_2}{T_1}. \end{aligned} \quad (18)$$

The recovery time T_r is the duration for $w(t)$ to increase from w_0 $(\approx \sqrt{2B_K})$ to $w(t_e)$, thus

$$T_r = ((B_1 + \frac{T_2}{T_1}) - \sqrt{2B_K})T_1. \quad (19)$$

The number of transmitted TCP packets in this period is $\frac{1}{2}((B_1 + \frac{T_2}{T_1})^2 - 2B_K)$, and the total average throughput is

$$Th = \frac{(T_c - T_o - T_r)\mu + \frac{1}{2}((B_1 + \frac{T_2}{T_1})^2 - 2B_K)}{T_c} \quad (20)$$

Please refer to Sec. III for the meaning of T_c and T_o.

VI. Numerical Results and Discussions

In all simulations reported here, TCP-Reno is assumed, a TCP packet is chosen as the unit, and the error model in Section 2 is adopted so that the maximum throughput is 0.75.

First, Fig.7 shows that with ACK reservation scheme, no extra time is needed to wake up TCP source after the link outage is over (compared with Fig. 3), which means the recovery time is significantly reduced by using ACK reservation scheme.

Secondly, we focus on the case with the buffer size B less than B_{min}. Table I ($K = 1$ means ACK reservation is not used) shows that our analytical results match simulation results quite well. Furthermore, our proposal achieves nearly 50 percent improvement in terms of throughput, compared with the scheme without ACK reservation.

Finally, we evaluate the throughput of our proposal with different buffer capacity and different ACK reservation threshold.

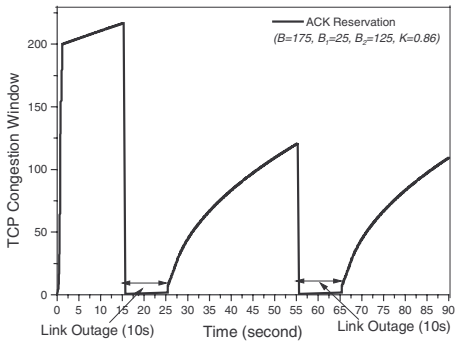

Fig. 7. Tracing TCP Congestion Window

(B, K)	T_r (second)		Th/μ	
	Simulation	Analysis	Simulation	Analysis
(175, 0.86)	2.44	2.4	0.730	0.732
(175, 0.95)	2.67	2.7	0.726	0.724
(200, 0.75)	2.15	2.1	0.739	0.740
(200, 0.85)	2.33	2.3	0.7364	0.732
(175, 1)	11.6	N/A	0.492	N/A
(200, 1)	11.6	N/A	0.495	N/A

TABLE I

RECOVERY TIME AND THROUGHPUT ($B_1 = 25, B_2 = 125, B_{min} = 319$)

Given $\lambda = 2500\,packet/s$, $\mu = 250\,packet/s$, $T_1 = 50ms$, and $T_2 = 500ms$, we get the value of B_1 and B_2 by using $B_1 = \mu T_1$ and $B_2 = \mu T_2$. The corresponding B_{min} is 183. From Fig.8, we see that the ACK reservation scheme can improve the throughput, approaching the maximum. Moreover, it is clearly observed that only the curve with $B = 185$ can reach the maximum 0.75 around the point $K = 0.88$, which matches very well the results from Eq.13 and Eq.11. ($B_{min} = 183$, $B_K = \frac{B_1+2}{8}B_1 = 23, \quad K = 1 - \frac{B_K}{B} = 0.88$)

VII. OTHER RELATED WORKS

Actually, the similar problem of long-time link outage also occurs in a cellular communication system due to hand-off. Some wireless TCP approaches have been proposed in the literature to solve such problem (e.g. M-TCP [4], W-TCP [5], etc.).

M-TCP [4], which is also based on TCP splitting, makes use of the persist mode of TCP. At the split point of the TCP connection (or proxy), an ACK for all but the last byte is forwarded to the fixed host. On detection of a link failure, the TCP layer at the mobile is frozen. The proxy on receiving no ACKs from the mobile, advertises zero window along with the last byte, also called ZWA (Zero Window Advertisement), thus putting the fixed host in persist mode. When the link is up, the proxy receives an ACK from the mobile, resulting in the proxy informing the fixed host of the true window and thus restarting TCP. But, the loss of ACK packets (due to congestion or unreliable link) will have great adverse impact on M-TCP, since the task of *freezing* and *restarting* TCP is accomplished with *one* ACK packet (while our scheme employs a burst of ACK to restart TCP, which is more robust to ACK packet loss). Furthermore, M-TCP needs modifications in TCP layer at both proxy and receiver.

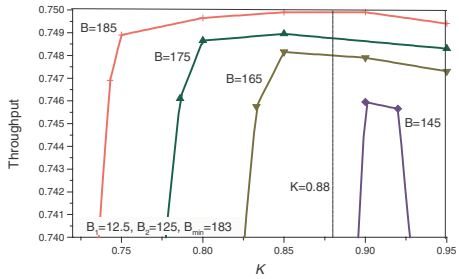

Fig. 8. The effect of buffer size B and the threshold K for ACK reservation on the throughput

W-TCP [5] is a new transport protocol designed specifically for wireless wide-area network using *selective acknowledgment* and no retransmission timers for loss recovery; this enables efficient loss recovery without going into prolonged timeouts in the worst case. Therefore, the problem of long-time link outage has little impact on the performance of W-TCP. But W-TCP needs modifications in TCP layer at both sender and receiver.

Compared to M-TCP and W-TCP, the ACK reservation scheme is implemented only at the satellite gateway (or proxy) with no need to modify the TCP stack in the end systems (sender or receiver), significantly reducing the complexity of implementation. Furthermore, it is designed on *cumulative acknowledgment mechanism* and *additive increase multiple decrease (AIMD) window control mechanism*, the two basic functions adopted by most popular TCP versions (e.g. Tahoe, Reno and NewReno). Therefore, it can be used with the above TCP versions as well.

VIII. CONCLUSION

In this paper, we studied the problem of TCP splitting in satellite networks. To alleviate the problem of slow TCP source recovery from link outage , we propose a simple but effective solution based on *ACK reservation*. By storing some ACK packets in advance, and sending them back after the ending of link outage, the sender's TCP window can be forced to increase more rapidly. Furthermore, we derive the minimum buffer requirement for the maximum performance. The optimal value of the key parameter K is also given. Simulation validates our analysis, and shows that ACK reservation improves TCP end-to-end throughput significantly in the presence of long-time link outage.

REFERENCES

[1] A. V. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP", IEEE Trans. Computer, vol. 46, no. 3, pp. 260-278, Mar. 1997.
[2] G. Fairhurst, L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", draft-ietf-pilc-link-arq-issues-03.txt, Aug. 2001.
[3] J. Ishac, M. Allman, "On The Performance of TCP Spoofing in Satellite Networks", MILCOM 2001.
[4] K. Brown and S. Singh, M-TCP: TCP for Mobile Cellular Networks, ACM Computer COmmunications Review 27, pp. 19-43, Oct. 1997.
[5] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, V. Bharghavan,"WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks". Wireless Networks 8, pp. 301-316, 2002.