TCP OVER SATELLITE LINK WITH SACK ENHANCEMENT

Ghanthiya Leerujikul* and Kazi M. Ahmed
Telecommunications Program
School of Advanced Technologies
Asian Institute of Technology

Abstract

This paper uses simulations to study how different kinds of TCP congestion avoidance algorithms behave on the satellite link. GEO satellite is assumed. For better understanding, TCP over terrestrial network is first discussed. Then, in the case of TCP over satellite link, TCP with varying advertised window, *cwnd*, size is employed to observe the behavior of each TCP congestion avoidance algorithms. Our results reveal that SACK outperforms the other congestion avoidance schemes significantly. For the case of TCP SACK over satellite, the size of the advertised window greatly affects its performance, especially when the window size exceeds the TCP standard window size.

1. Introduction

Since nowadays the need for high-mobility remote access broadband services like satellite networks are becoming more popular, Transmission Control Protocol (TCP) is among the schemes that are being studied in order to provide reliable data delivery over it. TCP itself also provides various congestion avoidance schemes that may be used over the satellite link. However TCP may have to face with some difficulties due to the satellite link characteristic like large link delay. Therefore, the study of congestion avoidance of TCP over satellite link is an important issue.

There are many previous studies considering the performance of congestion avoidance of TCP. However, most of them considers TCP in terrestrial network like as mentioned in [4]. Currently, efforts are underway to enable better utilization of satellite links. Many studies have proposed techniques to improve the TCP performance over satellite links as stated in [6,11]. Throughput performance of TCP over satellite has been mentioned in [12]. However, the authors did not consider the performance in terms of percentage efficiency and queuing delay, neither did they emphasize on any TCP congestion avoidance schemes. The behavior of each TCP schemes with varying advertised window size is not mentioned as well. In this paper, we study various congestion avoidance schemes of TCP, including Tahoe, Reno, New Reno and SACK in both terrestrial and satellite link scenario. The performance in terms of throughput, percentage efficiency and mean queuing delay are discussed. Moreover the behavior of each scheme with varying window size is also described.

2. Tahoe TCP

Tahoe algorithm includes Slow Start, Congestion Avoidance, and Fast Retransmit. With Fast Retransmit, when the sender receives three duplicate acknowledgments for the same TCP segment, it concludes that the segment indicated by the ACKs has been lost, and immediately retransmits the lost segment.

3. Reno TCP

The Reno TCP implementation is the enhancement to Tahoe by modifying the Fast Retransmit operation to include Fast Recovery. In Fast Recovery, instead of Slow Starting as is performed by a Tahoe TCP sender, the Reno sender inflates its *cwnd* by the number of additional duplicate ACKs it has received to clock subsequent outgoing packets. When an ACK for new data is received, the sender exits Fast Recovery.

4. New-Reno TCP

In New-Reno TCP, there is a small change to the Reno algorithm at the sender that eliminates Reno's wait for a retransmit timer when multiple packets are lost from a window. The change concerns the sender's behavior during Fast Recovery when a partial ACK is received. In New-Reno, partial ACKs do not take TCP out of Fast Recovery as in Reno. Instead, partial ACKs received during Fast Recovery indicate that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New-Reno can recover without a retransmission timeout, by retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. It remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged [4].

5. SACK TCP

TCP with Selective Acknowledgments (SACK TCP) has been proposed to efficiently recover from multiple segment losses. SACK TCP acknowledgment contains additional information about the segments that have been received by the destination. When the destination receives out-of-order segments, it sends duplicate ACKs (SACKs) acknowledging the out-of-order segments it has received. From these SACKs, the sending TCP can reconstruct information about the segments not received at the destination. When the sender receives three duplicate ACKs, it retransmits the first lost segment, and inflates its *cwnd* by one for each duplicate ACK it receives. This behavior is the same as Reno TCP. However, when the sender, in response to duplicate ACKs, is allowed by the window to send a segment, it uses the SACK information to retransmit lost segments before sending new segments. Therefore the sender can

* Currently Ghanthiya Leerujikul is working toward her Ph.D. at Assumption University, Thailand.

0-7803-7080-5/01/$10.00 ©2001 IEEE

recover from multiple dropped segments in about one round trip. While *cwnd* is inflating, TCP is sending missing packets before any new packets.

6. The Satellite Delay Calculation

The bandwidth-delay can be used to determine the suitable advertised window size used in data transmission. The round trip delay can be calculated with the equation given as

$$\text{Round trip delay} = \frac{2d}{c} \qquad (1)$$

where d is the slant range or the distance from the earth station to the satellite and c is 3×10^8 m/s, the speed of light [1].

The slant range can be obtained from equation 2. The value for the Earth radius r_e is 6370 km and r_s is the orbital radius. In this paper, GEO satellite is used therefore the orbital radius r_s is 42,242 km.

$$d = r_s\left[1+\left(\frac{r_e}{r_s}\right)^2 - 2\left(\frac{r_e}{r_s}\right)\cos(\gamma)\right]^{\frac{1}{2}} \qquad (2)$$

The central angle γ is related to the earth station north latitude L_e and west longitude l_s and the subsatellite point west longitude l_s by

$$\cos(\gamma) = \cos(L_e)\cos(l_s - l_e) \qquad (3)$$

For geostationary, the maximum slant range is d = 41,127 km. As a result, the roundtrip delay will be about 0.274 seconds.

7. Methodology

Results concerning on SACK performance as well as other congestion avoidance schemes are obtained from the simulation using Network Simulator or NS [5]. The simulation is divided into two cases: TCP over terrestrial network and TCP over satellite network. Some simulation parameters are taken from [4]. We use a finite-buffer droptail gateway for both cases. For the case of TCP over satellite network, the source and the receiving hosts communicate via GEO satellite. Therefore the distance between the source and satellite and the receiving host and satellite is fixed. The links is labeled with their bandwidth capacity, 1.536Mbits/s, and delay is varied according to *cwnd* size. With this amount of link capacity, the standard TCP window size is limited to 128 packets. The results obtained from the simulation are used to evaluate their performances. Furthermore, the value of *awnd* is varied in order to compare the performances for each type of TCP congestion avoidance schemes with varying window.

8. Performance Evaluation for TCP over Terrestrial Network

In TCP over terrestrial network, two scenarios are considered: one packet drop and three packet-drops. In the case of TCP with one packet drop, Tahoe introduces the highest mean queuing delay, least throughput and least percent efficiency when comparing to other algorithms. Once the drop occurred in Tahoe, congestion window is reset to one and transmits the dropped packet. After that congestion window is increased by one segment for every round trip time. This will take some round trips to efficiently utilize the network. TCP SACK seems to have the best performance. Although SACK obtains the same average throughput and percentage efficiency as Reno and New-Reno, it reveals the least mean queuing delay. This is because SACK can immediately retransmit the dropped packet when it received the ACK containing SACK option from the receiver.

Not much distinct performance evaluation is noticed under a single packet-dropped scenario. However, better results are obtained with more number of packets drop.

Figure 1 reveals TCP efficiency of the four congestion avoidance algorithms in a three packet-drops scenario. Here, Reno introduces the lowest throughput and lowest efficiency. In the same time it significantly obtained the highest queuing delay among the other three congestion avoidance schemes. This is because Reno needs to wait for timeout and it can retransmit at most one dropped packet per round-trip time. As a result, other congestion avoidance schemes are sending packets while Reno needs to wait for timeout. When comparing to the previous case, Reno significantly improves upon the behavior of Tahoe TCP when a single packet is dropped, but it can suffer from performance problems when multiple packets are dropped from a window of data like in this case.

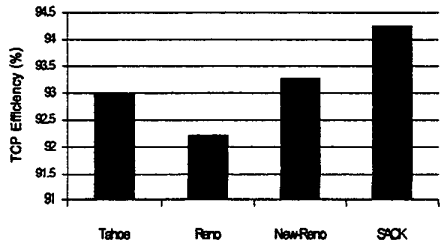

Figure 1 TCP Efficiency in Three Packet-Drops Case

The performance of SACK TCP is becoming more significant with the increasing number of packet drops from a window of data. Results show that SACK introduces the highest throughput as well as the highest percent efficiency. This is because SACK often prevents the need for a timeout and can recover quickly from multiple packet losses. As a result, it consumes the least time spending in the queue.

9. Performance Evaluation for TCP over Satellite Link

This section describes how different types of TCP congestion avoidance schemes react to multiple packet-drops that occur over satellite link. The performance evaluation is observed according to different sizes of TCP advertised window. With respect to these varied window sizes, the average queuing delay, number of packet drop, TCP efficiency and average throughput

will be observed. How SACK reacts to the exceeding standard TCP window size is also mentioned.

Results reveal that TCP SACK significantly outperforms the other three types of TCP congestion avoidance scheme. In a contrary way, New-Reno introduces the worst performance. This is because it can retransmit at most one dropped packet per round-trip time and must remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged.

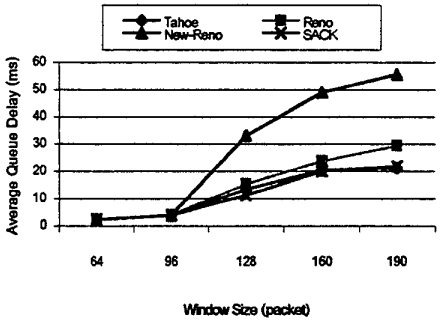

Figure 2 Average Queue Delay versus Window Size

The average queuing delay is plotted against the window size in Figure 2. All four TCP congestion avoidance schemes have similar reaction to the time spending in the queue when the window size is not fix. The graph reveals that the queue delay increases with the increasing size of TCP *awnd* window. Since the expansion of window size implies that more data packet is introduced into the network at a given time. As a result, more packet drops will occur as shown in Figure 3. In order to recover from this multiple packet-drops condition is also time consuming. Therefore a packet has to spend more time in the queue when the size of the window increases.

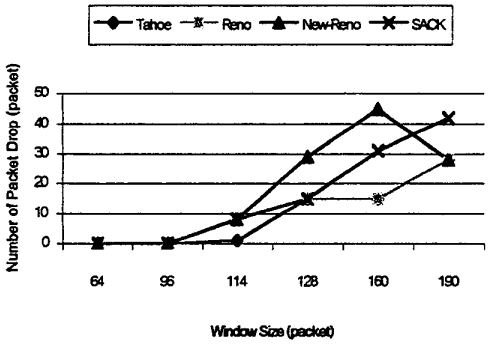

Figure 3 Number of Packet Drop versus Window Size

Refer to Figure 3, we can see that the number of packet drop increases with the increasing window size for TCP Tahoe, Reno, and SACK, but New-Reno tends to decrease the dropped packet when window size is 190 packets. However New-Reno is also expected to behave in the same way as the other three types. The reason why the number of packet drop for New-Reno decreases is because the simulation time used does not cover up to the time that all of the data outstanding when fast recovery was initiated has been acknowledged. To confirm this situation, Figure 5 reveals that New-Reno introduces the lowest throughput when window size is 190 packets.

Figure 4 shows TCP percentage efficiency plotted against window size. The graph reveals that efficiency tends to drop due to increase *awnd* size and this window size does not exceed 128 packets. However after this limitation, the efficiency tends to increase at window size of 160 packets and drops again at the size of 190 packets. However the throughput at window of 190 is more than the throughput at window of 160. This is due to what we have discussed previously that the number of packet drop increases with the increasing window size. Therefore, after exceeding window size of 128 packets, the throughput is also decreased.

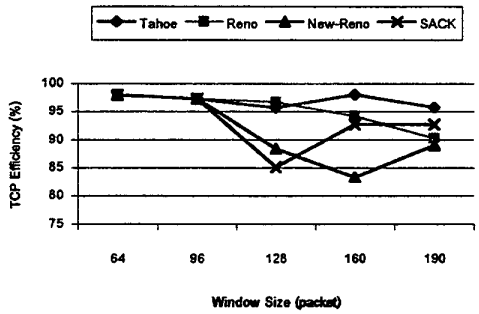

Figure 4 TCP Percentage Efficiency versus Window Size

Now let us consider the average throughput results from the varied window size as shown in Figure 5. We can see that the highest throughput occurred at window size of 96 packets, which is equal to the value of bandwidth delay product. The simulation results reveal the RTT of 255.3ms. Therefore we can calculate the suitable window size by:

$$\text{Window Size (bits)} = 1.536\times10^{6} \times 0.2553$$
$$= 392140.8 \text{ bits}$$
$$\text{Window Size (packets)} = 95.737 \text{ packets}$$

At this size of window, the maximum throughput is obtained. When window size is larger than 96 packets, the throughput starts to decrease.

From the graph, we can see that SACK recover no slower than Tahoe. This is because the larger window size introduces more losses. If more than half of the *cwnd* is dropped, then there will not be enough duplicate ACKs for PIPE (communication path) to become large enough to transmit any segments in the first RTT. Only the first dropped segment will be retransmitted on the receipt of the third duplicate ACK [9]. In the second RTT, the

partial ACK corresponding to the retransmitted packet will be received. As a result, PIPE will be decreased by 2 so that 2 packets can be sent. Therefore PIPE will double every RTT, and SACK recover no slower than slow start. However SACK is still be advantageous because timeout would still be avoided unless a retransmitted packet were dropped.

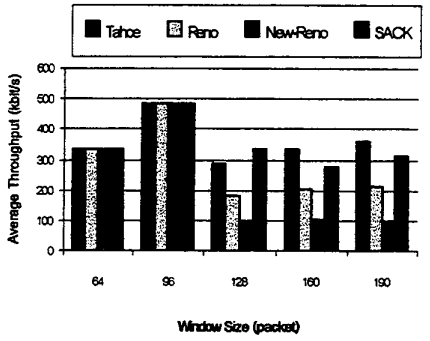

Figure 5 Average Throughput versus Window Size

10. Conclusions

The four TCP congestion avoidance schemes that this paper covers are Tahoe, Reno, New Reno and especially on SACK. In the terrestrial network case, results reveal that SACK does not provide any distinct improvement over the other three algorithms in the case of single packet drop. However when more packet-drops in a window of data occurs, TCP experiences poor performance. The cumulative acknowledgement in Tahoe, Reno, and New-Reno can only notice the sender about a single lost packet per round trip time. SACK, on the other hand, can overcome this limitation by using SACK options sent from the receiver to inform the sender of data that has been received. The sender can then retransmit only the missing data packet. Therefore SACK significantly outperforms the other three algorithms when multiple packets drop occur. It shows the highest throughput and TCP efficiency of 94%, and the lowest mean queuing delay.

Under satellite network scenario, again, TCP SACK significantly outperforms the other three types of TCP congestion avoidance scheme, whereas New-Reno introduces the worst performance. This is because New-Reno can retransmit at most one dropped packet per round-trip time and must remains in fast recovery until all of the data outstanding when fast recovery was initiated has been acknowledged. When window size is taken into account, the results show that the advertised window greatly affects TCP efficiency, average throughput, and mean queuing delay. By examining the performance evaluation results, we realize that the highest throughput occurs when the window size equals to the calculated bandwidth-delay product.

From all that we have stated above, we conclude that TCP SACK improve the performance of TCP, not only over terrestrial link, but also over satellite link. For the case of TCP SACK over satellite, the size of the advertised window greatly affects its performance, especially when the window size exceeds the TCP standard window size. However, SACK would still be advantageous because timeout would still be avoided.

References

[1] **Ahmed, K. M.**, (1999). Satellite Communications, Lecture Notes on Course No. AT05.22. *Telecommunication Program, School of Advanced Technologies, Asian Institute of Technology, Bangkok, Thailand*, 1999.

[2] **Barakat Chadi, Chaher Nesine, Dabbous Walid and Altman Eitan,** (1999). Improving TCP/IP over Geostationary Satellite Links, *Global Telecommunications Conference-Globecom'99*, Italy, pp.781-785.

[3] **Caceres, R.**, and **Iftode, L.**, (1994), The effect of mobility on reliable transport protocols, Proceedings of the 14th International Conference on Distributed Computing Systems, June 1994, pp 12-20.

[4] **Fall Kevin and Floyd Sally**, (1996). Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, Vol. 26, July 1996, pp. 5-21.

[5] **Fall Kevin and Varandhan Kannan**, (2000). *ns* Notes and Documentation, *The VINT Project, February* 25, 2000.

[6] **Ghani Nasir and Dixit Sudhir,**(1999). TCP/IP Enhancements for Satellite Networks, *IEEE Communications Magazine*, Vol. 377, July pp. 64-72.

[7] **Glover R. Daniel and Kruse Hans**, (1998). TCP Performance in a Geostationary Satellite Environment. *Annual Review of Communications 1998, International Engineering Consortium*, April 1998.

[8] **Goyal Rohit, Jain Raj, Kalyanaraman Shivkumar, Fahmy Sonia, Vandalore Bobby, Kota Sastri**, (1997). TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks, *Proc. ICCCN97*, Las Vegas, September 1997 pp17-27.

[9] **Jain Raj, Kalyanaraman Shiv, Goyal Rohit, Fahmy Sonia, and Kim Seong-Cheol**, (1997). UBR+: Improving Performance of TCP over ATM-UBR Service, *Communications ICC'97 Montreal, Towards the Knowledge Millenium, IEEE International Conference*, Vol. 2, pp.1042-1048.

[10] **Mathis Matthew, Mahdavi Jamshid, Floyd Sally, and Romanow Allyn**, (1996). TCP Selective Acknowledgment Options, *Internetdraft RFC 2018*, October.

[11] **Metz Christopher,** (1999). TCP Over Satellite... The Final Frontier, *IEEE Internet Computing*, Vol. 31, January-February pp.76-80.

[12] **Partridge Craig and Shepard Timothy J.**,(1997). TCP/IP Performance over Satellite Links, Vol. 115, *IEEE Network*, September/October pp.44-49.