ENSC 835: NETWORK PROTOCOLS AND PERFORMANCE

CMPT 885: SPECIAL TOPICS: HIGH-PERFORMANCE NETWORKS

TCP performance analysis over satellite links

Spring 2002

Final Report

Kenny, Qing Shao

Grace, Hui Zhang

{qshao, hzhange}@cs.sfu.ca

Department of Engineering Science, Simon Fraser University

Abstract:
In this report, we present the simulation result of a comprehensive performance study of the Transmission Control Protocol (TCP) enhancements in the satellite environment. Our results illustrate many of the remaining challenges facing TCP over satellite links and also provide demonstration of three techniques that can be used to optimize performance in long delay networks. We also evaluate the different TCP flavors under the satellite error prone links, and explore the bottleneck buffer effect on end-to-end TCP connections.

Index

1. Motivation

Satellite communication technology has been developed for nearly 50 years. Over the past few years, the demand to use satellite devices to access the Internet is growing because satellite communication can deliver Internet services to consumers and institutions in remote areas of the world not covered by good terrestrial connectivity. In addition, satellite broadcast system is ideal for multicast service, satellite access method is also adapt for asymmetric internet data transmission, service providers can use satellite ocean range beam easily extend their network nationwide and ocean wide without last mile problem. In the mean time, operation expense is not related to the distance. Due to the above reason, satellite technology will be more utilized in Internet transmission. It is expected that TCP protocol will be frequently used over the Satellite network in near future.

Our works are important for several reasons. First, commercial satellite companies (e.g., Loral, Hughes, Lockheed Martin) have announced plans to build large satellite system to provide broadband data service. Second, our simulation can help researchers to understand how satellite links characteristics affect TCP performance. Third, Some of the TCP problems experienced on satellite links today will rise in future high-speed terrestrial networks because of the similar bandwidth-times-delay property. Problems like large window size, prolonged slow start period, and ineffective bandwidth adaptation affect both networks. They place satellite networks and gigabit terrestrial networks in the same class of extensions for better performance.

2. Introduction to satellites communication

2.1 Satellite links characteristics

The following two channel characteristics will greatly affect the TCP performance.

2.1.1 Delay:

Most commercial satellite systems work in geosychonous equatorial orbit (GEO) today. A GEO satellite is located directly above the equator, exactly 35,800 kilometers out in space. At that distance, it takes the satellite a full 24 hours to circle the planet. Since it takes Earth 24 hours to spin on in its axis, the satellite and Earth move together. So, a satellite in GEO always stays directly over the same spot on Earth (Figure2.1).

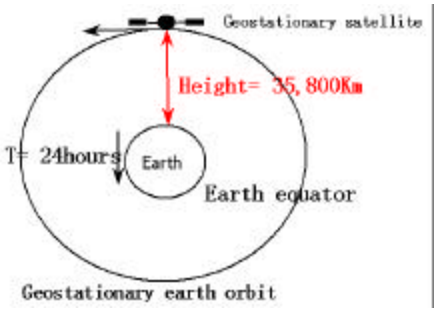

Figure 2.1 Geosychonous equatorial orbit satellite

Because the distance between GEO satellite and earth station is rather long, propagation delay is the dominant part compared with the transmission delay and queuing delay. If an earth station transmits a message to another earth station through GEO links, the propagation delay will typically be 280ms (one way) of fixed delay. If the acknowledgement (ACK) is also returned through satellite channel, the Round Trip Time (RTT) will be around 560ms. This delay will greater than the average terrestrial transmission delay, which cause larger round trip time and bandwidth product.

2.1.2 Transmission errors:

Like other wireless communications, satellite communication exhibits higher error characteristics compared to wired links. Commercial satellite systems usually work at Ku (12GHz-14GHz) band, so that they can offer higher bandwidth with smaller antenna. But Signal working in this frequency suffers environmental impairments such as rain and fog; the bit error rate (BER) will increase from 10^{-7} to 10^{-3} in this condition.

Satellite networks are not the only environments where the above two characteristics are found. The mechanisms discussed in this report should benefit most networks, especially those with the above characteristics. (E.g., gigabit networks have large bandwidth-delay product)

2.2 Two types of GEO satellites

There are two types of GEO satellites in industry; we will investigate TCP performance in both types.

2.2.1 Bent pipe Satellite

Bent pipe GEO satellites act as a relay station in space. People use them to bounce messages from one part of the world to another. Signal is amplified and retransmitted but there's no improvement in C/N ratio, since there's no demodulation, decoding or other type of processing.

2.2.2 On board Processing satellite

Satellite performs tasks like demodulation and decoding which allow signal recovery before retransmission. Since the signal is available at some points in base band, other activities are also possible, such as routing, switching, etc. Our TCP burst simulation was based on this type of satellite.

3. TCP Improvement for satellite link

The introduction of the high delay and error prone satellite link into a network, using protocols primarily designed for terrestrial networks, creates significant performance degradation. Quite a lot of research has been performed with regard to TCP performance in satellite environments, either in the frame of research teams [10] or separately by individual researcher; an extensive review of this is given in [6]. The vast majority of the solutions suggested concern the TCP redesign implementation. We used ns-2 [13] to simulate three redesigning TCP extensions, which are large window size, large initial window size and maximum segment size.

The performance of three redesigned TCP extensions was examined on the bent pipe satellite in the scenario depicted in Figure 3.1.

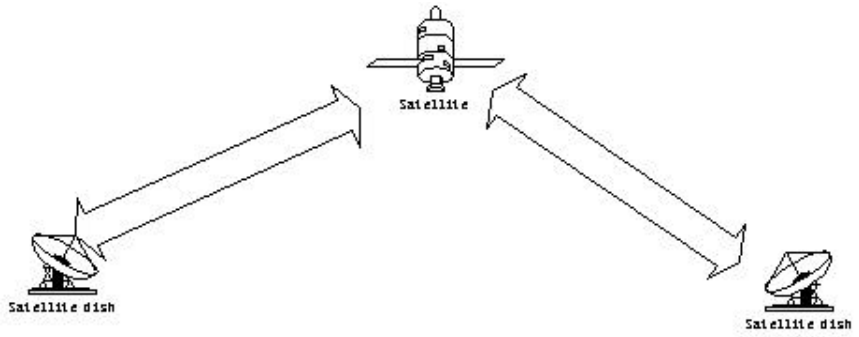

Figure 3.1 Simulation scenario

The simulation was based on the abstract implementation of duplex T1 connections. In order to simply the result, we use the default value (1000K byte) as packet size. Since selective acknowledgement (SACK) option is currently being used in many popular operating systems and is expected to be widespread soon, all of simulations were based on SACK. In the end, we also compare three TCP versions Reno, SACK and Vegas in section 3.4.

3.1 Large window size

The original TCP standard limits the advised receive windows by only assigning 16bits of header space for the value. Thus the advised windows can be no more than 64KB. Since the maximum throughput of a TCP connection is bounded by the RTT (RFC 793), as seen in the formula:

$$\text{throughput} = \frac{\text{congestion window size}}{\text{round trip time}}$$

Without larger window size, the TCP connection over a GEO system will be limited to throughput:

$$\text{throughput} = \frac{64\text{Kbytes}}{560\text{ms}} \approx 117{,}027\ \frac{bytes}{\text{sec}cond} \approx 940\,kbps$$

This upper bound on TCP throughput is independent of the bandwidth of the channel [6]. A TCP connection running over a full T1 channel can only achieve a maximum throughput of approximately 940Kbps with a 64KB receive window. So larger windows size can allow TCP to fully utilize higher bandwidth links over long-delay channels such as those found in satellite links.

We varied the TCP window size from 16KB to 128KB with FTP application; evaluate the throughput since throughput determines the bandwidth utilization of the link from the system manager's point of view.

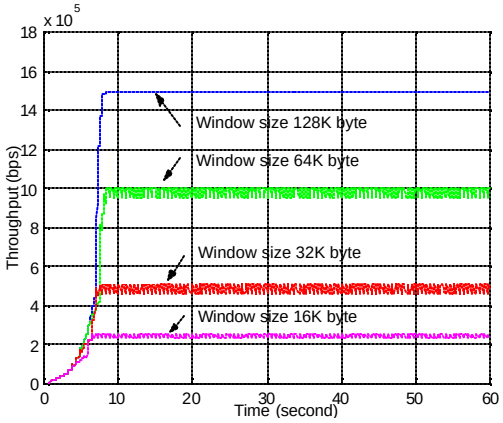

Fig 3.2.Throughput vs. time

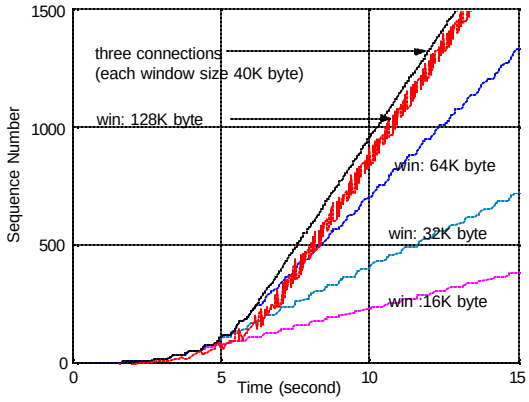

Fig 3.3. Sequence number vs. time

The relative efficiency of the different window size is clearly depicted in the representative diagram of figure 3.2. As the window size increases, the throughput of the connection also increases. When the window size is set to 128 KB, a single connection can fully utilize a T1 channel. From the simulation trace, we can find that the sequence number is not in order; this lead to the throughput fluctuates. An interesting phenomenon is that the packets arrive in order as when we choose 128KB as window size, this lead the throughput curve more flat than others.

We also established multiple FTP connections with small window size. Figure 3.3 shows that three connections with 40KB window size can achieve almost the same effect compared with single connection with the large window size (128KB). This result shows that if a satellite link was shared with many users, the large windows size might not be necessary. The service providers can still fully utilize their channel capacity if there are many long-lived connections with small windows size. Long-lived connection means that the bulk file transfers long enough so that it can fully utilize the large window size.

In practice, memory and operating system resource limit the window size. Because of historical implementation issues, the practical window size is often limited to 32KB.

RFC 1323 introduced an option for TCP to advertise windows to increase the maximum window size from 2^{16} to 2^{32}, allowing better utilization of links with large bandwidth delay products. To obtain good TCP performance over satellite links, both sender and receiver use this extension; applications should also set the size of the send and receiver buffers to be bandwidth times delay.

3.2 Large initial windows size

The slow start algorithm is used to gradually increase the size of TCP's congestion window (cwnd). The algorithm is an important safe guard against transmitting an inappropriate amount of data into the network when the connection starts up [1]. However, slow start can also waste available network capacity, especially in long-delay networks. Slow start is particularly inefficient for transfers that are short compared to the bandwidth-delay product of the network.

Telnet application is a typical example with a transactional behavior. This interactive application often opens a TCP connection only to send a small amount of data. Under standard TCP, even a small transaction must undergo the slow start procedure, so this application is very inefficient for satellite networks.

Several proposals have suggested ways to make slow start less time consuming. One method that will reduce the amount of required time by slow start is to increase the initial value of cwnd [5]. By increasing the initial value of cwnd, more packets are sent during the first RTT of data transmission, which will trigger more ACKs, allowing the congestion window to open more rapidly.

We used Telnet application to investigate this approach to mitigate the underutilization of the network during the slow start phase of a TCP transfer.

As the official minutes of the TCP Implemention Working Group meeting show, at the December 1997 meeting at the Washington IETF, there was rough consensus to allow an initial window of two segments. At the March 1998 meeting at the L.A. IETF, there was a rough consensus to allow an initial window of three or four packets (depending on the segment size) for experimental purposes. Therefore, we choose initial window size 1, 2, 3 and 4 segments respectively. Since we are only interested in the slow start period, so we just ran our simulation for one minute.

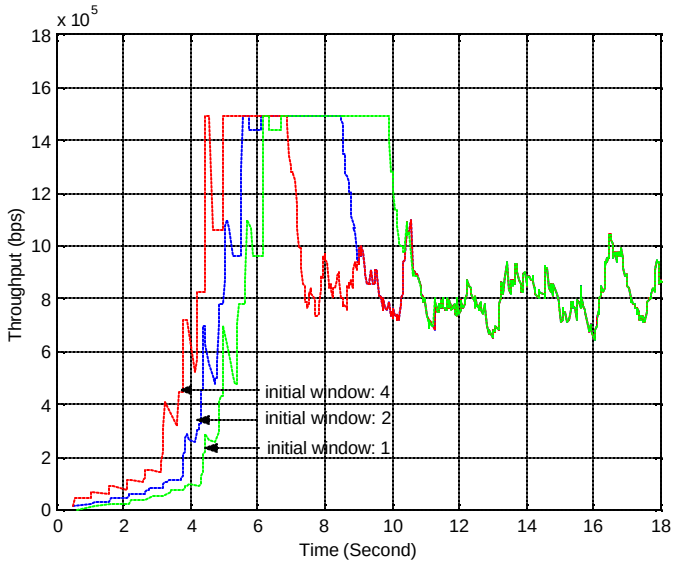

Figure 3.4 Throughput vs. time

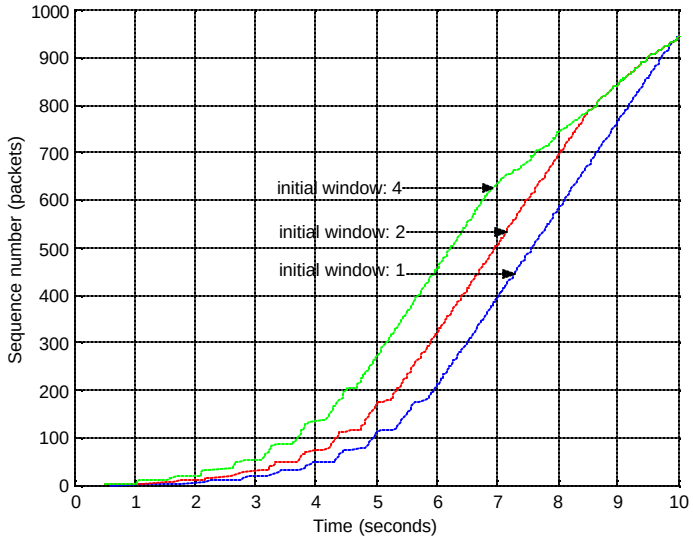

Figure 3.5 Sequence number vs. time

As shown in figure 3.4, the slow start period decreases as the initial value of cwnd is increased. From figure 3.5, we can observe that impact is especially significant for short transfers. For instance, it takes 5 seconds to transmit 300 packets with initial window size 4. But those data will take around 6 seconds to be delivered with standard TCP, which will last 25% longer. We can also infer that the impact for the longer transfer is much less due to the relatively short amount of time spent using slow start when compared to the total time required transfer the file.

3.3 Maximum segment size

Maximum segment size defines the largest segment of TCP data that can be transmitted.

It can be described as the following equation:

$$MSS = MTU - TCP\ header - IP\ header$$

Here, MTU means maximum transmission unit (or maximum packet size).

The segment size is considered to have a direct impact on the TCP performance, and some related researches have been proposed such as Path MTU Discovery to encourage TCP to use largest possible packet size [6].

As we have discussed in the previous part, increasing TCP's initial window size is based on it's segment size, hence, we want to verify if using larger segment size can allow TCP sender to increase the congestion window size in terms of bytes rapidly. Therefore, it may significantly improve the throughput to fully utilize the channel.

We assumed that both TCP sender and receiver have agreed on sending large segment size, and packets are not being fragmented. With the same scenario as figure 3.1, we simulated FTP application with window size 128KB. Since packets size 576 KB is widely used in the network, we tried its multiple size (576, 1152, and 1728 packets) to run the simulation.

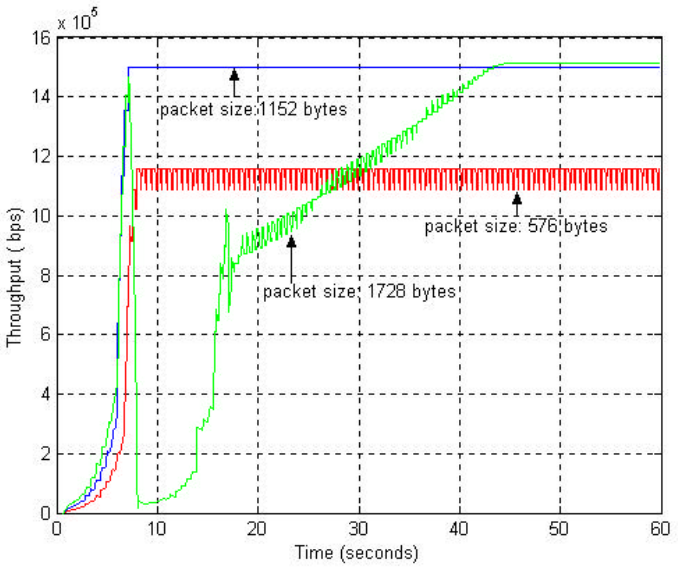

Figure3.6: Throughput vs. time (Different packet size)

From the figure 3.6, we found that if we used the packet size of 1152 bytes, we can improve the throughput more than 20% compared with the packet size of 576 bytes. Transmitting with 1152 bytes per packet can almost fully use the T1 channel, however, the packet size of 576 bytes can't.

Also, we could infer that when transmitting a certain amount of data, larger packet size may achieve better performance. Because it reduces the number of headers and condenses the cost of routing decisions, both of which are dealt with per-packet rather

than per byte.

We found that the packet size of 1728 bytes performs abnormally. During slow start period, the congestion window doubled every round trip time. However, due to the large window size (128KB) and the large segment size, its transmission rate exceeds the given bandwidth 1.544Mbps, then caused the severe congestion in the channel. Therefore the throughput decreased rapidly and it reentered the slow start period. In a word, using large maximum segment size can improve the throughput; on the other hand, too large MSS may cause congestion. That the larger MSS is the better performance it can achieve is not always true, thus choosing a suitable MSS should be very careful.

3.4 Comparison of different TCP flavors

TCP Reno was widely used in the industry for years. As more and more people focus on Internet research, more advanced TCP extensions were developed. The SACK and Vegas option have been approved as a proposed standard within the IETF as RFC 2018 [11] and RFC [3], and are expected to enjoy wide deployment.

TCP Reno is designed to handle packet loss by identifying and re-sending lost segments; however, Reno assumes that source of all packet loss is network congestion. Consequently, Reno invokes congestion control, reduces its congestion windows. Therefore, its transmission rate is a result of any packet loss [1]. This response is inappropriate when faced with loss due to corruption rather than congestion, while this is often the case on satellite communication links. Reno's congestion control algorithm works well in dealing with congestion-induced loss, but only results in reduced throughput without providing any benefits. . In addition, Reno sender can only be notified at most a single loss in the receiver's buffer because the ACK numbers is decided by the highest in order sequence number. Therefore, sender will wait till to time out, and then begin slow start procedure when more than one packet drop.

RFC 2018 introduced a selective acknowledgement (SACK) option to TCP. Using SACK option, receivers can inform senders exactly which segments have arrived, rather than replying on TCP's cumulative acknowledgement. This allows a TCP sender to efficiently recover from multiple lost segments without reverting to using a costly retransmission timeout to determine which segments need to be re-transmitted [11]. It also hasten recovery and prevent the sender from becoming window limited, thus allowing the pipe to drain while waiting to learn about lost segments. This ability is of particular benefit in keeping the pipe full and allowing transmission to continue while recovering from losses [4].

Unlike TCP Reno, TCP Vegas adopts different mechanisms to detect packet losses and available bandwidth. It controls cwnd by estimating RTT and calculating the difference between the expected flow rate and the actual flow rate [3]. It linearly increase adjusts

TCP's congestion window upwards or downwards, so as to consume a constant amount of buffer space in network switched. It detects packets loss earlier than Reno and uses a slower multiple decrease than Reno. The TCP Vegas eliminates the need to tune the receive window to serve as an upper bound on the size of the congestion window. It can avoid network congestion without overdriving the link to find the upper bound, even when operating with large windows. TCP Vegas increases its cwnd more slowly than Reno and by measuring the achieved throughput gain after each increase to detect the available capacity without incurring loss.

We compared SACK, Reno, and Vegas under the congested long-delay satellite link, using FTP application in the same scenario as figure 3.1. For each of these simulations, the queues in routers were set above the delay bandwidth product, meaning that there was enough buffering in the routers so that no TCP segments were lost due to congestion.

Figure 3.7 shows the throughput performance of SACK, Reno and Vegas as a function of bit-error rate

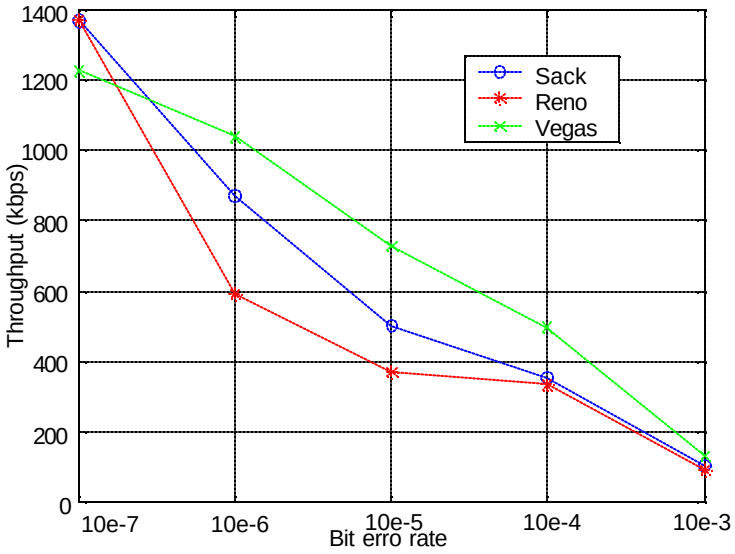

Figure 3.7 Throughput vs. bit error rate. (SACK, Reno, Vegas)

For bit error rate of 10^{-7}, the SACK performance almost the same as the Reno, we think that this is because there isn't many multiple drop in this good condition. For bit error rate of 10^{-6} and 10^{-5}, SACK is shown to greatly improve throughput relative to TCP Reno. Reno learns about segment loss slowly, at most one per round-trip, because it lacks a selective acknowledgment mechanism. For bit error rate of 10^{-4}, the channel is too noisy for either protocol to exhibit good performance.

Vegas always performs better than SACK and Reno except when BER is 10^{-7}. It increase 40% in throughput compared with Reno, this verified what the Vegas's authors has claimed in his proposal [3]. Vegas' improved throughput is primarily a consequence of congestion avoidance. Through its congestion avoidance, Vegas drops fewer packets, so it experience fewer timeouts, thus retransmits fewer segment than Reno. For larger window sizes, Vegas' congestion avoidance decreases the chance of multiple segment drops. Also, Vegas recovers from multiple segment drops better than Reno. We are surprising that its throughput is lower when BER is 10^{-7}. This might because in a low bit error channel, due to its conservative characteristic, TCP Vegas might gain a little lower throughput compared with SACK and Reno.

Overall, How to optimize algorithms which governs use of the SACK information is still the subject of research, however the basic algorithms are now beginning to be implemented. Vegas has not been widely implemented and is not universally accepted by the Internet community and is still a subject of much controversy.

4. TCP burst

4.1 Introduction:

Next generation GEO satellites will be more advanced and complex, which will carry an on board switch to facilitate the telecommunication network. They will form links between satellites, provide more flexible access to the end users. Those on board switch will act as a router in the terrestrial network. Since it is impossible to arrange big components in the spacecraft, those on board switches will not have powerful function as the routers in the terrestrial networks do. Therefore, except for high delay and error prone characteristics of satellite links, TCP will encounter a new challenge: limited buffer size.

TCP is burst in nature, which is due to the burst data transmission. It may lead to poor performance because of the limited buffer size. In order to meet better utilization of the high delay bandwidth product channel, large window size is recommended. However, this results in large burst data added to the network in a short interval so that creates long queues in the router.

We study TCP burst in order to find out how it influences the TCP performance, and we hope our simulation could explain TCP burst over most high delay-bandwidth product links rather than focus on the satellite links.

In addition, in our simulation, we tried two types of acknowledgments: basic acknowledgment and delayed acknowledgment. Delayed acknowledgment allows TCP transmit an ACK for every second full-sized data segment received. If a second data segment is not received within a given timeout, an ACK is transmitted [9]. This mechanism is widely deployed in real TCP implementations.

4.2 Simulation:

a. Simulation scenario:

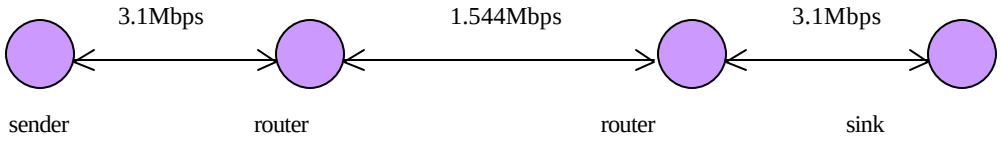

Figure 4.1 Simulation scenario

b. Some important parameters:
- Window size: 128 packets.
- Use various buffer sizes to run the simulation with basic acknowledgment and delayed acknowledgment respectively.

c. Simulation Result:

- Using basic acknowledgment:
  I. Queue length vs. time with various buffer sizes:

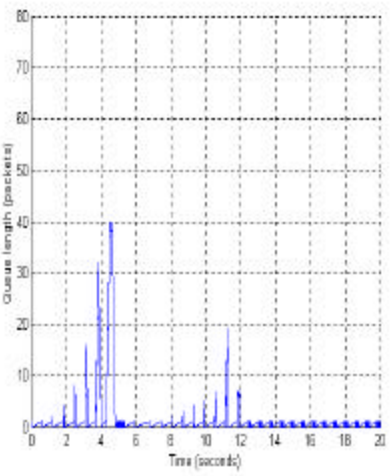

Figure 4.2
(Buffer size: 40 packets)

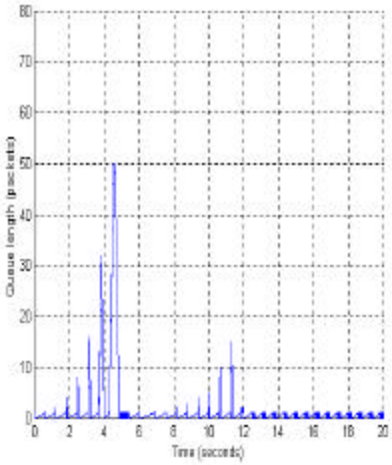

Figure 4.3
(Buffer size: 50 packets)

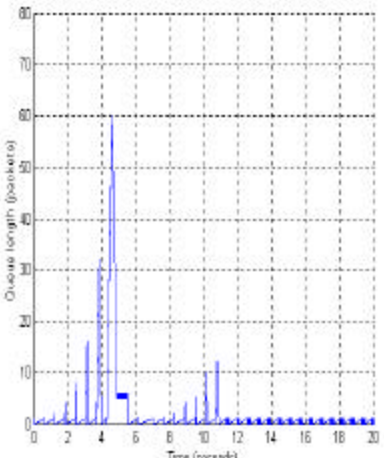

Figure 4.4
(Buffer size: 60 packets)

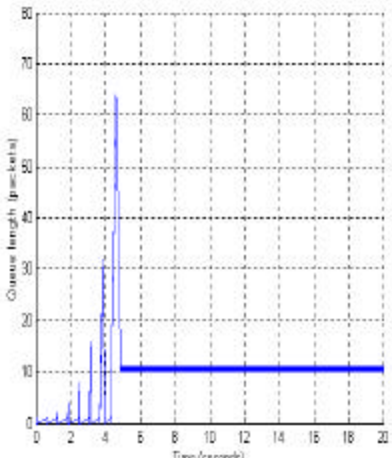

Figure 4.5
(Buffer size: 70 packets)

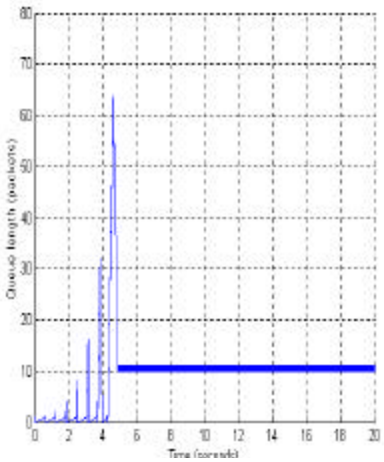

Figure 4.6
(Buffer size: 80 packets)

Figure 4.2 to 4.6 show the queue length in the bottleneck with different buffer size. All of these figures have a common characteristic: the exponential growth of the queue length during the initial period. It represents TCP burst in slow start. After slow start the queue occupancy decreases. Also, according to the simulation result, there shows little difference between the buffer size of 70 packets and 80 packets.

II. Sequence number vs. time with various buffer sizes:

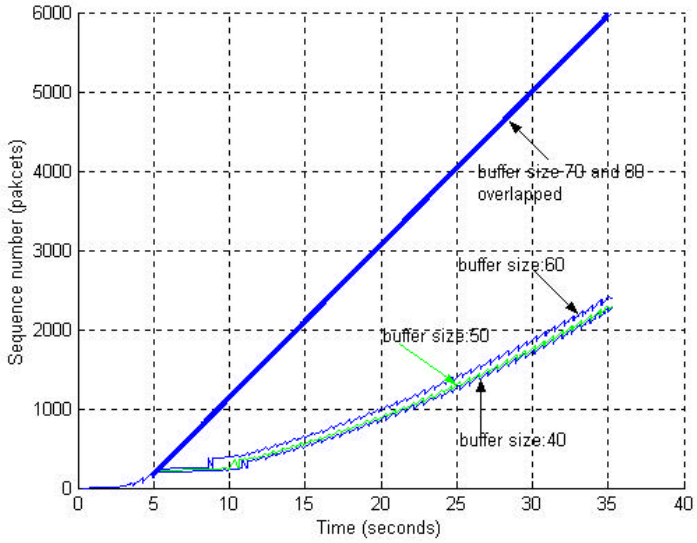

Figure 4.7 Sequence number vs. time (Different buffer size)

The buffer size has severe impact on TCP performance, however, when the buffer size reaches certain amount, it doesn't influence the throughput any more. The simulation results are quite similar when we use the buffer size of 70 packets and 80 packets.

- Using delayed acknowledgment:

I. Queue length vs. time with various buffer sizes.

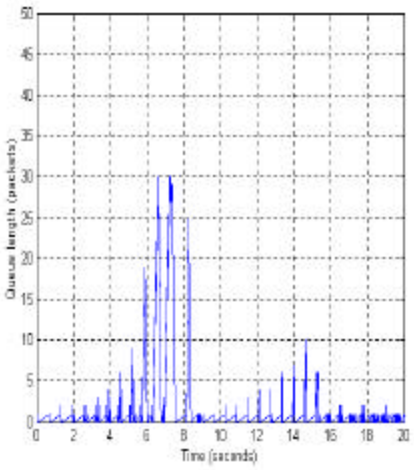

Figure 4.8
(Buffer size: 30 packets)

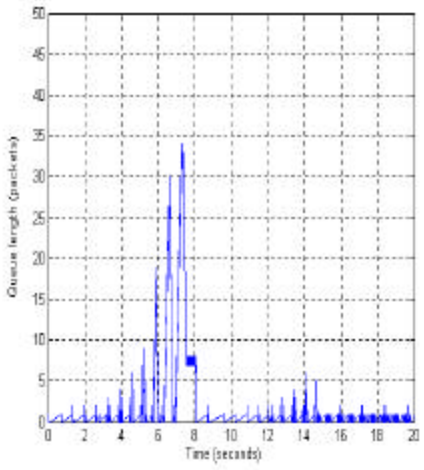

Figure 4.9
(Buffer size: 34 packets)

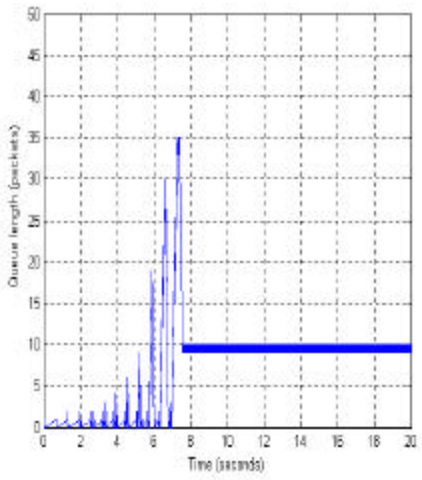

Figure 4.10
(Buffer size: 40 packets)

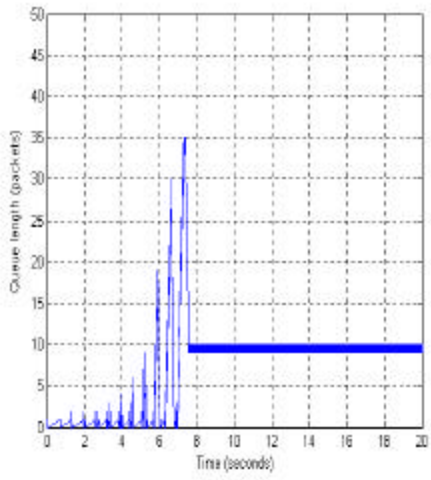

Figure 4.11
(Buffer size: 50 packets)

From all of the figures above, TCP burst is also significant during slow start period. Moreover, TCP burst almost remains the same when using the buffer size of 40 packets and 50 packets.

II. Sequence number vs. time:

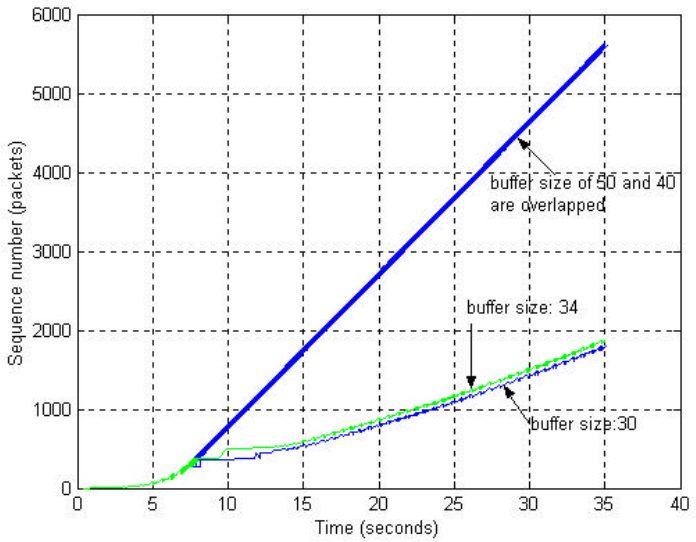

Figure 5.12 Sequence number vs. time (Different buffer size)

In our simulation, the buffer size of 40 packets and 50 packets performs almost the same.

4.3. Analysis:

Some macro burst can be observed during slow start period. This might be explained as following:
All the Acks arrive at the beginning of every RTT, and then TCP sender doubles the window size if all the ACKs have received. This causes a large amount of data transmission in a short interval, followed by a long idle time especially in a long delay network.

TCP burst has significant impact on TCP performance. Insufficient buffer size will cause the throughput degraded severely: Burst of TCP would lead to packets drops because of the overload of the buffer. Moreover, the burst nature of traffic will increase the queuing delay, which may results in extending RTT. Both of these factors would directly influence the throughput.

Due to the different ACK mechanism, delayed ACK is less aggressive, so it shows less burst compared with the basic ACK algorithm. As a result, it requires less buffer size. However, at the same period of time, delayed ACK may achieve a little lower throughput than basic ACK.

In a single connection, when using basic acknowledgment, the ideal router buffer size should be at least half of the maximum window size (Wmax). During slow start period, as the TCP sender receives one acknowledgment, it increases the window size by one, and then inject a pair of (two) packets into the network. The router server only has time to deal with one of the two packets before the next pair arrives; hence, one of these packets accumulates in the bottleneck queue for every pair of packets arriving. Accordingly, if the TCP sender increases the window size from Wmax/2 packets to Wmax packets, the bottleneck queue has to store Wmax/2 packets or some of the packets may drop. Therefore, the ideal router buffer size should be at least half of the maximum window size. When using delayed ACK, generally, the TCP sender increases the window size about 3 times every two round trip time. With the same logic, the ideal bottleneck queue size should be at least one third of the maximum window size (Wmax/3).

From our simulation result, increasing the buffer size neither alleviates TCP burst nor improves the throughput if the buffer size is already larger than Wmax/2 using basic ACK (or larger than Wmax/3 if using delayed ACK). This may help choose a suitable buffer size of routers.

Some solutions have been proposed such as paced TCP and pacing ACK, both of which are trying to alleviate the TCP burst during slow-start with even smaller buffer size. These algorithms may contribute to next generation GEO satellite networks and other links, which are suffered from the bottleneck (limited buffer size) and burst traffic.

5.Conclusions

5.1 Conclusion

From the theoretical analysis and simulation, our study shows that three redesigning TCP extensions have positive effect in the large bandwidth-delay product environment.

- Larger windows size option can greatly improve the single connection throughput, which is limited by the standard TCP. Multiple long-lived connections can also fully utilize the channel capacity with small window size.
- Larger initial window size will be particular benefit for Telnet application in satellite links.
- Suitable maximum segment size will improve the throughput to fully utilize the bandwidth.
- SACK and Vegas have better performance than Reno dose when satellite links is in high bit error rate conditions.
- TCP burst will significantly influence the TCP performance: With insufficient buffer size, the throughput will degrade severely. The ideal buffer size of the bottleneck should be Wmax/2 if using basic ACKs or Wmax/3 if using delayed ACKs

5.2 The difficulties and what was learned

The difficulty of this project is how to implement ns-2 satellite model, how to set up the related parameters and design the simulation.

From this project, we are familiar with the ns-2 simulator, awk function, and tcl/tk programming. This will greatly help in our future research.

5.3 Future work

In the simulation, we showed that the larger window size and initial window size extensions should be adopted, but there are some uncertainties need to be addressed.

Large window size can lead to more rapid use of the TCP sequence space. Therefore, along with large window size, Protect Against Wrapped Sequence Number algorithm (PAWS) is required.

Larger initial windows size may increase packet drop rate for a network with a high segment drop rate. So how to choose suitable initial window size is still an open question.

Furthermore, in this project, we simulate and analyze the behavior of TCP burst. The future work can focus on how to solve this problem. Some new algorithms have been

proposed to alleviate TCP burst with limited buffer size such as paced TCP and pacing ACK.

In the long term, further improvements can be made at the application protocol level by extending the current TCP standard. People are exploring Http1.1 for web browse application and Xftp for FTP application in long-delay network. Much work needs to be done on possible extensions to ensure that they do not negatively affect the Internet as a whole.

6. Reference

[1] V. Jacobson, "Congestion Avoidance and Control", In ACM SIGCOMM, 1988.
[2] J. Mo, R. J. La, V. Anantharam, and J. Warland, "Analysis and comparison of TCP Reno and Vegas", Proceedings of the Conference on Computer Communications (IEEE Infocom), New York, Mar. 1999.
[3] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance", In Proceedings of the SIGCOMM '94 Symposium (Aug. 1994) pages 24-35.
[4] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP", SIGCOMM Computer Communication Review, 26(3), July 1996.
[5] Joanna Kulik, Robert Coulter, Dennis Rockwell, and Craig Partridge,"A Simulation Study of Paced TCP", September 1999.
[6] [RFC 2488] "Enhancing TCP over Satellite Channels", Mailman, and D.Glover, January 1999.
[7] Larry L.Peterson and Bruce S.Davie, "Computer networks: A system approach." Morgan Kaufman, 1996.
[8] [RFC2581] [8] [RFC 2581] "TCP Congestion Control", M. Allmanm, V. Paxson, W. Stevens, April 1999.
[9] [RFC 1122] "Requirements for Internet Hosts—Communication Layers", R.Braden, October 1989.
[10] [RFC 1323] "TCP Extensions for High Performance", V.Jacobson, R.Braden, D.Borman, 1992.
[11] [RFC 2018] "TCP Selective Acknowledgment Option", M. Mathis, J. Mahdavi, October 1996.
[12] [RFC 1191] "Path MTU Discovery", J. Mogul, S. Deering, November 1990.
[13] http://www-mash.cs.berkeley.edu/ns.