TCP Congestion Control in Wired and Wireless networks

Mohamadreza Najiminaini
(mna28@cs.sfu.ca)

Term Project ENSC 835 Spring 2008

Supervised by Dr. Ljiljana Trajkovic

School of Engineering and Science

Simon Fraser University

21th of April 2008

ABSTRACT

Over the past two decades, many congestion control algorithms have been proposed for wired and wireless networks in TCP. The main goal of all these proposed algorithms is to acquire better performance for throughput and good put in networks. In this project, we are going to simulate some of congestion control algorithms (Tahoe, Reno, New Reno, and SACK) in wired and wireless networks using Opnet modeler and see the performance analysis in terms of congestion window size and sent segment sequence number.

TABLE OF CONTENT

1 Introduction

TCP is the transport protocol exploited in the Internet for reliable data delivery and is vital for numerous key applications including WWW, e-mail and Telnet. The algorithm consists of three different phases: slow-start, congestion avoidance and retransmission/recovery. Each phase performs a part of TCP congestion control algorithms. All of the congestion control algorithms consist of the same mechanisms for slow-start and congestion avoidance phases but they may have differences in fast retransmit and fast recovery mechanisms. Hence, many literatures have been dedicated their work to compare various TCP congestion control algorithms in terms of congestion window size, download response time and sent segment sequence number.

In the literature [1], the comparison of TCP Tahoe, Reno, and SACK has been done by using ns-2 as a network simulator and it shows that SACK outperforms Tahoe and Reno in terms of congestion window size when multiple packet drops happen in a window. Also, in another literature [2], the modified Tahoe has been proposed for wireless networks and they have implemented wireless network scenarios in Opnet Modeler in order to compare TCP Tahoe and the modified Tahoe together in terms of congestion window size and download response time. Hence, they have acquired that the modified Tahoe has better performance than Tahoe in wireless networks.

In this research project, our aim is to simulate some congestion control algorithms in TCP (Tahoe, Reno, New Reno, and SACK) using Opnet Modeler simulation environment and we intend to compare the congestions control algorithms in terms of congestion window size and sent segment sequence number. Hence, in section 2, we explain about different congestion algorithms and mechanisms. In section 3, we show the simulation scenarios and compare

the algorithms in terms of congestion window size and sent segment sequence number in wired and wireless network topologies. In section 4, we conclude our research project.

2 Various TCP Algorithms and Mechanisms

2.1 Various Mechanisms in TCP

In this section, we explain about slow start, congestion avoidance, fast retransmit, and fast recovery mechanisms.

2.1.1 Slow-Start

To resolve congestion, the slow start algorithm has been proposed. The basic of this approach is the idea of a congestion window size (CWND). When the connection is established between a sender and a receiver, the CWND is set to one packet. When the packet receives to the receiver side, the receiver sends an ACK packet to the sender including the sequence number 1. At time of the receiving the ACK packet, the CWND is increased to two and the two packets are sent to the receiver. When the ACK3 receive to the sender side, the CWND becomes four and it sends four packets. The sender stops increasing the window size when CWND reaches the limit of the network capacity. The limit is defined as the minimum of window that sender can transmit and window that receiver can receive. figure 1 shows that the slow_start algorithm.

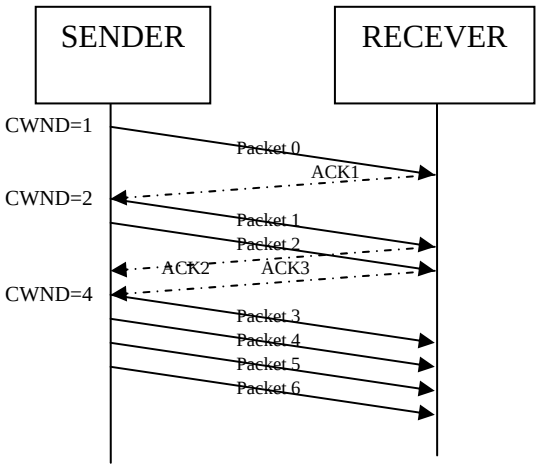

Fig.1. Slow-start mechanism

2.1.2 Congestion Avoidance

The congestion occurs when traffic rate at which packets arrive at routers is more than the routers can send. In general, there are two way indication of packet loss: a timeout happening and the receipt of duplicate ACKS.

Congestion avoidance and slow start algorithm work together to handle window size. These two algorithms define window size according to the minimum window size in the sender-side (CWND) and the receiver-side (RWND).

Window Size = min (RWND, CWND)

First of all, when the connection is established, the CWND is equal to one packet and it increases due to slow start algorithm until it reaches the ssthresh which usually is defined as 65536 bytes but if congestion happens then the value of the ssthresh is set to half of the CWND and CWND get equal to one packet. However, CWND increases exponentially with slow start algorithm; when it reaches the ssthresh, every time the sender gets an acknowledgment from the receiver it increases its window size linearly by this formula:

$$cwnd = cwnd + \frac{1}{cwnd}$$

2.1.3 FAST RETANSMIT

The old TCP detects the network congestion and lost packets by timeout mechanism. When the packet is sent, TCP sets up its own timer to retransmission timeout period (RTO) for this packet. When an immediate acknowledgment receives, the RTO will be expired for the packet. In the case of TCP does not receive ACK packet in RTO period, the sender will retransmit the packet whose

timer is expired. Besides, the CWND is set to one packet and ssthresh to (CWND old)/2.

Fast retransmit algorithm retransmits packet without waiting for retransmission timeout. In the case of packet n disorder in receiver, the sender receives the duplicate ACK n from receiver for the packet n then it sends packet n once again before the RTO is expired.

2.1.4 FAST RECOVERY

"After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached (3).

2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.

3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost."[3]. W. Stevens, "TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms,". Network Working Group, RFC2001, Jan 1997.

2.2 Congestion Control Algorithms

The first algorithm is the slow-start algorithm which is described before. The second one is the Thaoe's algorithm which operates as follows:

- After fast retransmit the TCP sets window size to 0 and sstresh to old window size/2.
- TCP starts alow start.
- When window size reaches ssthresh, TCP triggers to congestion avoidance.

The third one is Reno. In the comparison with above algorithm it has following differences:

- If the packet loss causes by RTO (congestion is serious) window size is set to one and do start slow.

- In the case that packet loss is indicated by duplicate ACK, congestion is not serious. It means at least the receiver successfully receives three packets. Then, congestion avoidance, not slow start is performed. Window size is set to old_window_size/2.

"The New-Reno TCP includes 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 that acknowledges some but not all of the packets that were outstanding at the start of that Fast Recovery period. In Reno, partial ACKs take TCP out of Fast Recovery by "deflating" the usable window back to the size of the congestion window. In New-Reno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication 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, retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged." [1] K. Fall and S. Floyd, "Simulation based comparisons of Tahoe, Reno and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, July 1996, pp. 5-21.

"The TCP selective acknowledgment mechanism (SACK) helps improve performance. The receiving TCP host returns selective acknowledgment packets to the sender, informing the sender of data that has been received. In other words, the receiver can acknowledge packets received out of order. The sender can then retransmit only the missing data segments (instead of everything since

the first missing packet). “[1] K. Fall and S. Floyd, “Simulation based comparisons of Tahoe, Reno and SACK TCP,” *Computer Communications Review*, vol. 26, no. 3, July 1996, pp. 5-21.

3 Simulation Scenarios

3.1 Simulation for Wired Scenarios

In order to investigate about the performance of the congestion control algorithms in wired networks in terms of congestion window size and sent segment sequence number, we have performed three simulation scenarios in Opnet Modeler.

As illustrated in figure 3, the first scenario consists of a point-to-point client to server connection and a packet discarder used to drop one packet within the simulation time. In this simulation scenario, the server supports a FTP application to provide a service for the client. The FTP application has been characterized as a constant file size of 1600000 bytes and a constant inter-request time of 3600. The amount of simulation time is 10 minutes and the actual time has prolonged around 5 minutes.

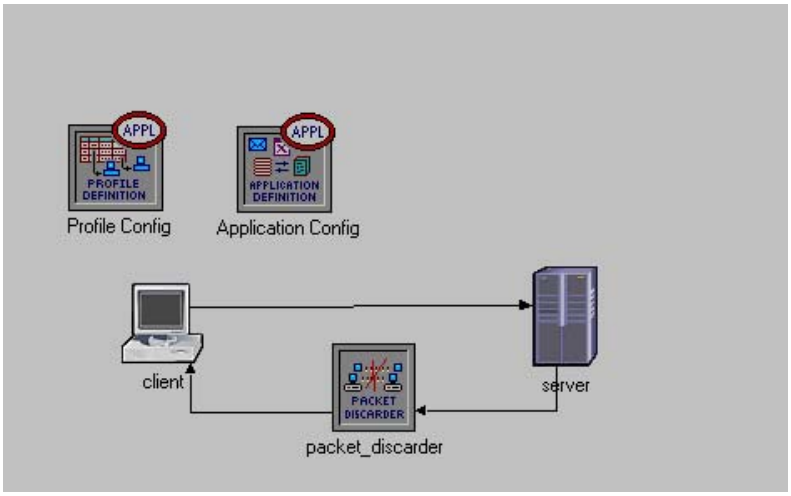

Fig.3. Point-to-Point client to server connection

According to the simulation results for the congestion window size as shown in figure 4 and 5, when a packet drop happens at 1 min and 47 sec, all congestion control algorithms behave in a different way. Thus, it is obvious from figure 4 and 5 that Tahoe performs slow start mechanism after setting its window size

and sshthresh when a packet drop occurs. However, Reno, NewReno and SACK perform their own fast retransmit and fast recovery algorithms respectively after a packet drop occurs. Also, it shows that all algorithms have the same download time in case of one packet drop.

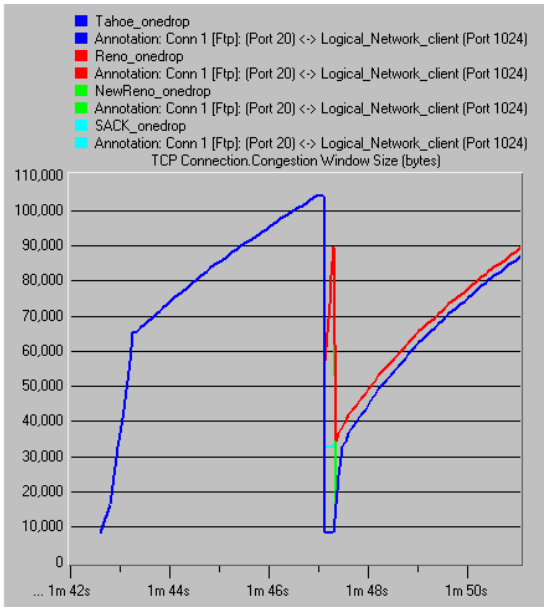

Fig.4. CWND in PPP

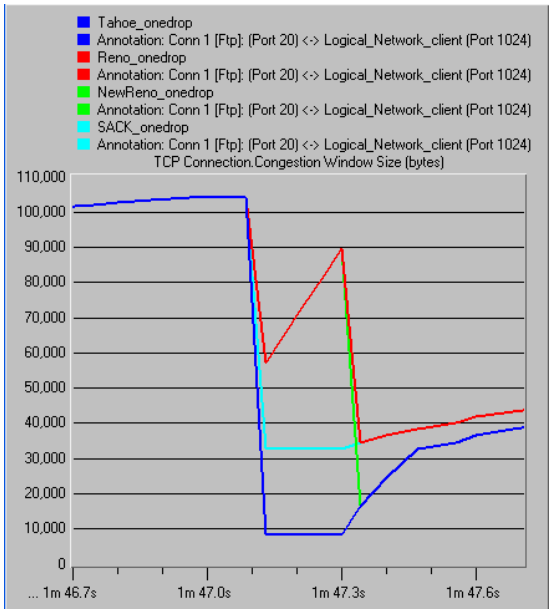

Fig.5. CWND in PPP (zoomed version)

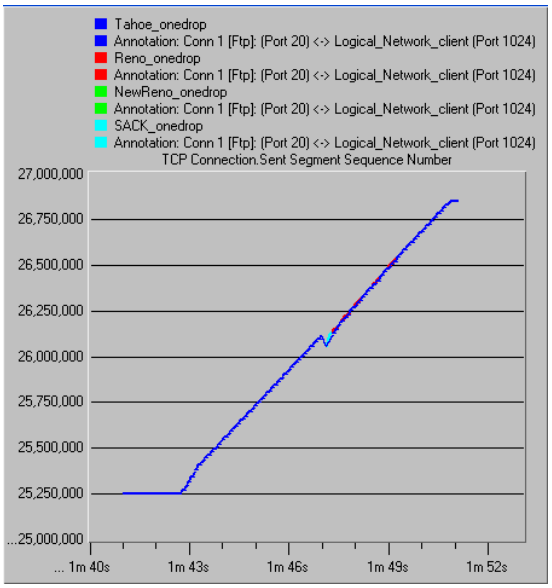

Fig.6. Sent segment sequence number

In case of sent segment sequence number simulation results, since only a single packet drop occurs, all TCP congestion control algorithms perform similar as shown in figure 6.

In order to examine congestion control algorithms more, we have performed another scenario same as a first scenario except that two packets are discarded in a window by the packet discarder. The congestion window size simulation results are shown in figure 7 and 8 for two packet drops.

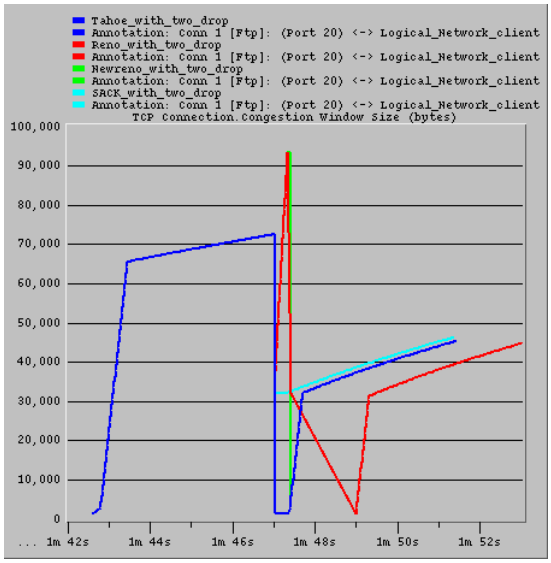

Fig.7. CWND in PPP (two drops)

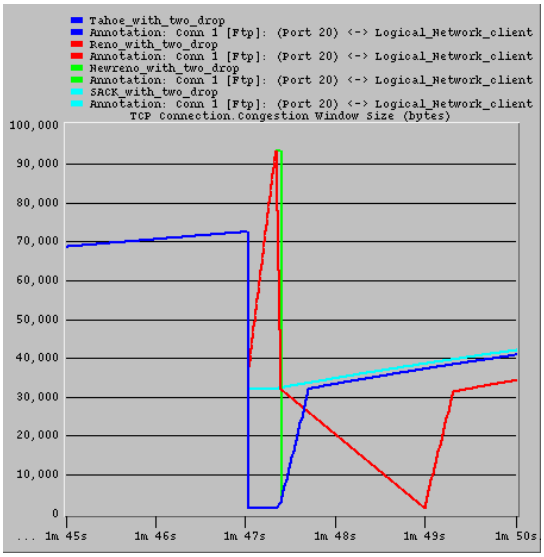

Fig.8. CWND in PPP (zoom version)

It turns out that Tahoe, NewRew, and SACK behave similar to one packet drop case. However, Reno can not handle two packet drops in a window and it performs slow-start when it exits from the fast recovery mechanism. In addition, as it is obvious from a figure 7, the download time from the server to the client for Reno algorithm takes longer than the other algorithms. The sent segment sequence number simulation results in case of two packet drops are similar for all algorithms as shown in figure 9.

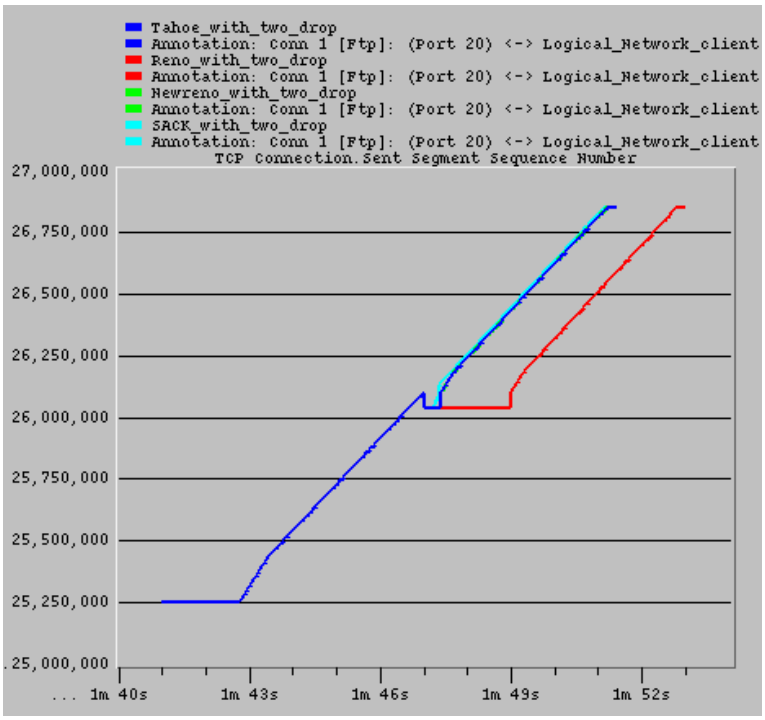

Fig.9. Sent segment sequence number two packet drops

Since congestion control algorithms in TCP have a huge effect on controlling congestion in internet and using a network bandwidth efficiently. Thus, we desire to accomplish a simulation scenario on WAN topology. Therefore, in this simulation scenario, we use USA map as the scale to implement our simulation scenario for the wired network. In the USA scale, we use two subnets, IP cloud (Internet), links, applications, and profiles as shown in figure 10.

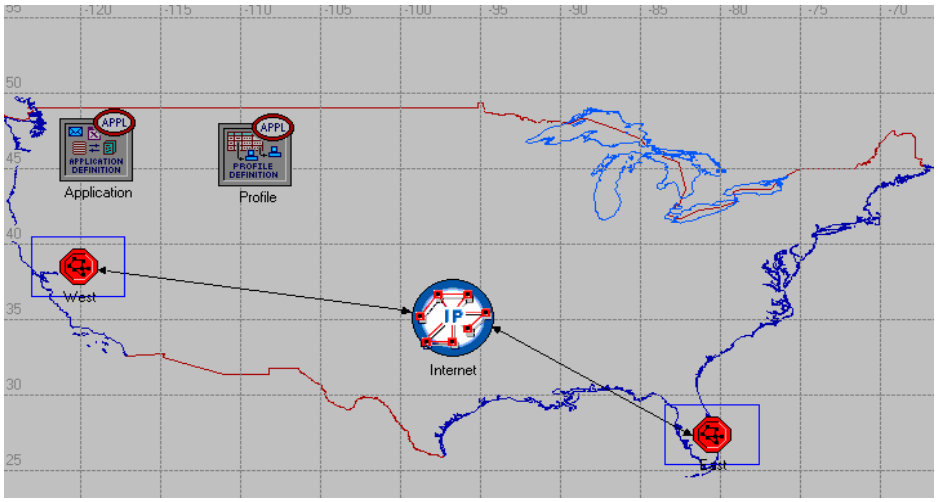

Fig.10. WAN topology

The links between IP cloud (Internet) and the subnets are DS3 link. It has two different subnets. The west subnet has two components as follows: a server and a router which is connected to the IP cloud as shown in figure 11. Also, for the east side, it has a router which is connected to IP cloud (Internet) and the workstation as shown in figure 12. Also, in order to discard packets and apply packet latency in WAN topology, we set packet discard ratio and packet latency to 0.05 percent and 0.001 second respectively in IP cloud attribute parameters.

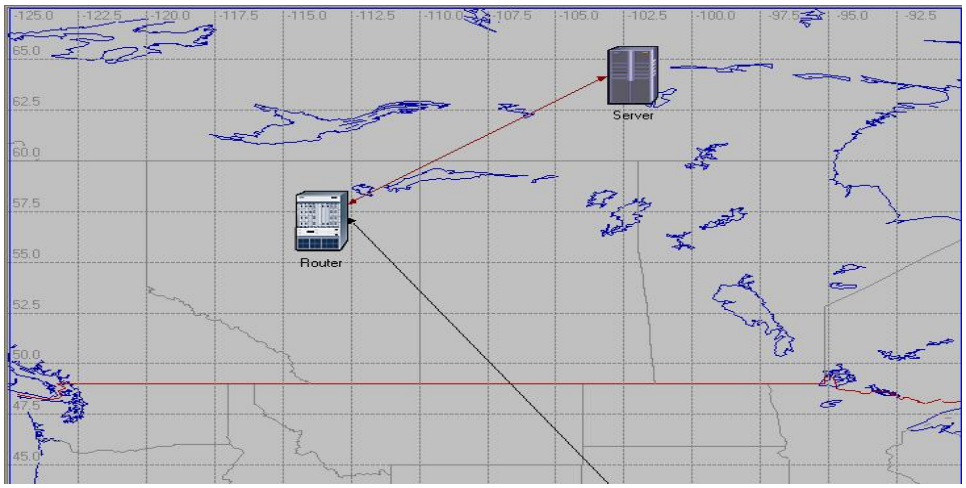

Fig.11. West subnet

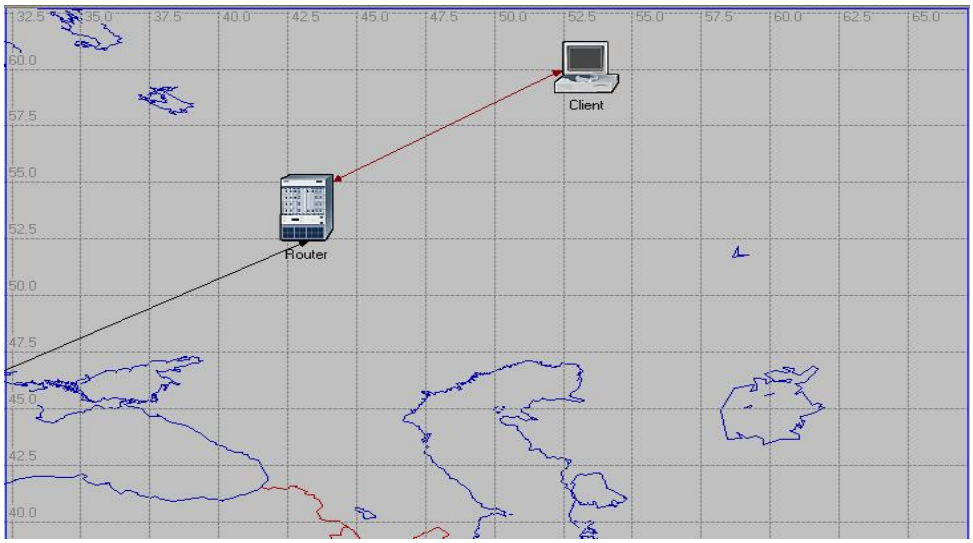

Fig.12. East Subnet

In this simulation scenario, the server supports a FTP application to provide a service for the client. The FTP application has been characterized as a constant file size of 1000000 bytes and a constant inter-request time of 3600. The amount of simulation time is one hour and the actual time has prolonged around 10 min.

In this simulation scenario, the distinction of congestion control algorithms becomes more obvious. As it is shown in figure 13, Tahoe is the worst case comparing to other algorithms and the required time to download the file has increased sharply. Besides, Reno download time is a bit more than SACK and NewReno. SACK and NewReno has the approximately same download time but it turns out that SACK does not have a sharp decrease when a packet loss occurs.

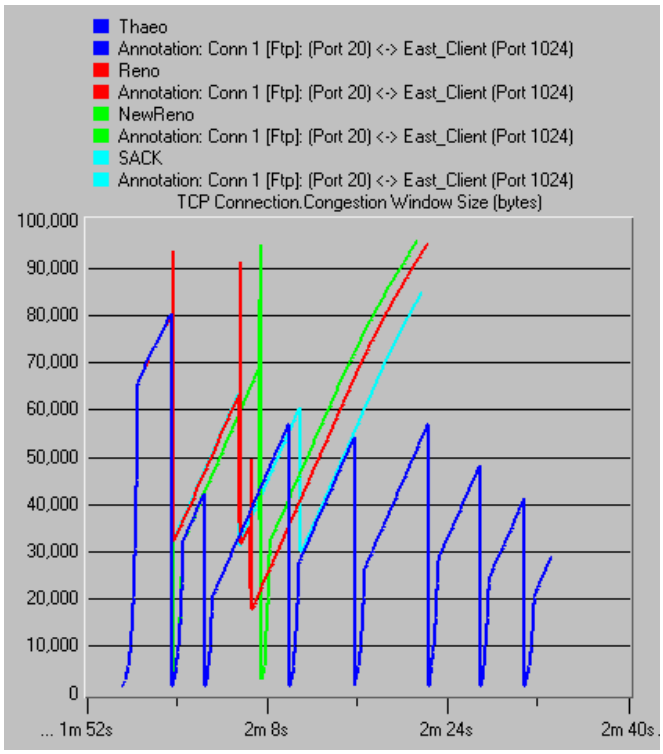

Fig.13. CWND in WAN topology

The sent segment sequence number simulation results are shown in figure 14. It shows that all algorithms reach to the same sent segment sequence number at

last and the only difference is download time for each algorithm as explained in congestion window size before.

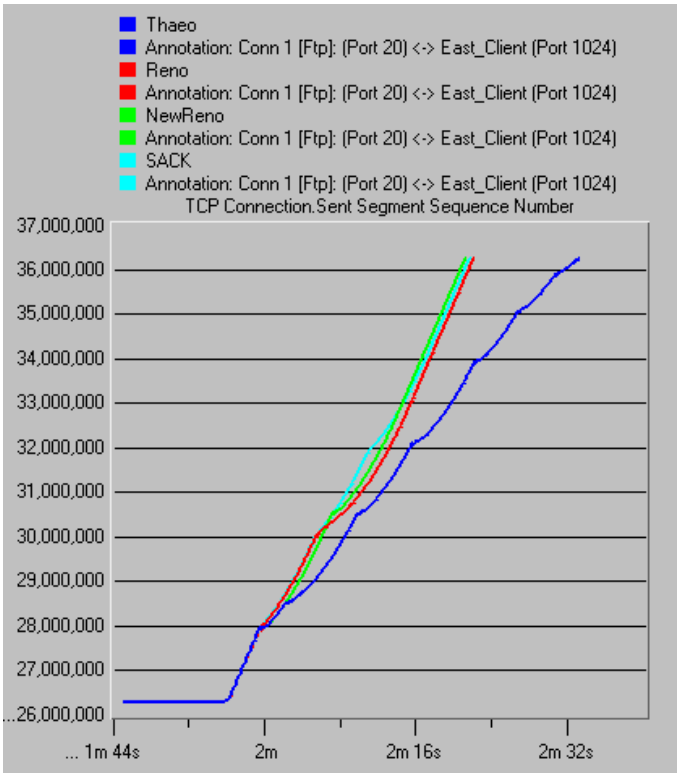

Fig.14. sent segment sequence number in WAN topology

According to all simulation results in wired network scenarios, NewReno and SACK outperforms Tahoe and Reno in terms of download time and congestion window size. However, sent segment sequence number is same for all algorithms in all simulation scenarios.

3.2 Wireless Simulation Scenario

In wireless simulation scenario, we desire to observe congestion window size for various TCP congestion control algorithms in case of signal attenuations which cause packet drops in a wireless environment. Thus, in this simulation scenario, a FTP server, one wired router, one base station which works as a wired and wireless router and a mobile wireless client are exploited as shown in figure 15. The FTP application has been characterized as a constant file size of 50000000

bytes and constant inter request time of 3500 sec. The wireless mobile client moves on the trajectory which is indicated by the green lines in figure 15. When the mobile client moves on the trajectory, beyond a certain distance from the base station, signal attenuations occur and the packets between the client and the base station start to be dropped.

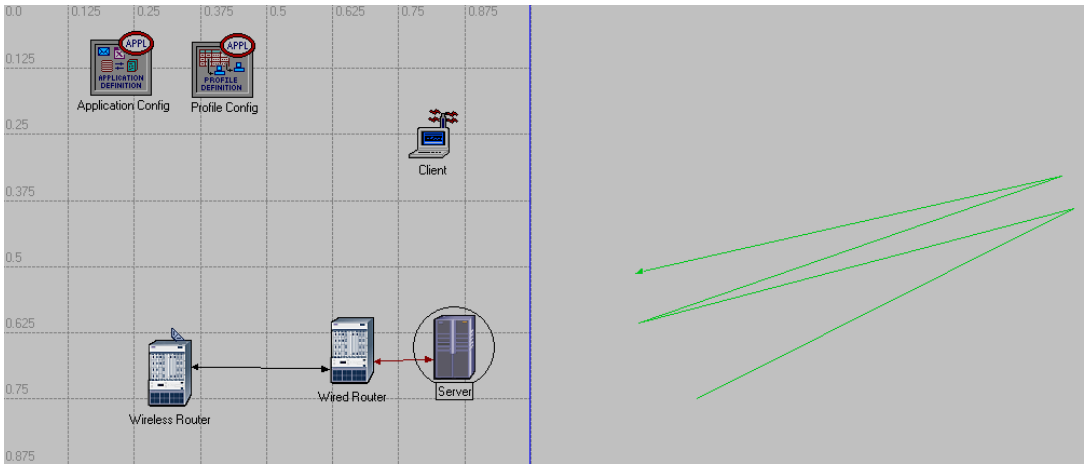

Figure 15. Wireless topology for simulation

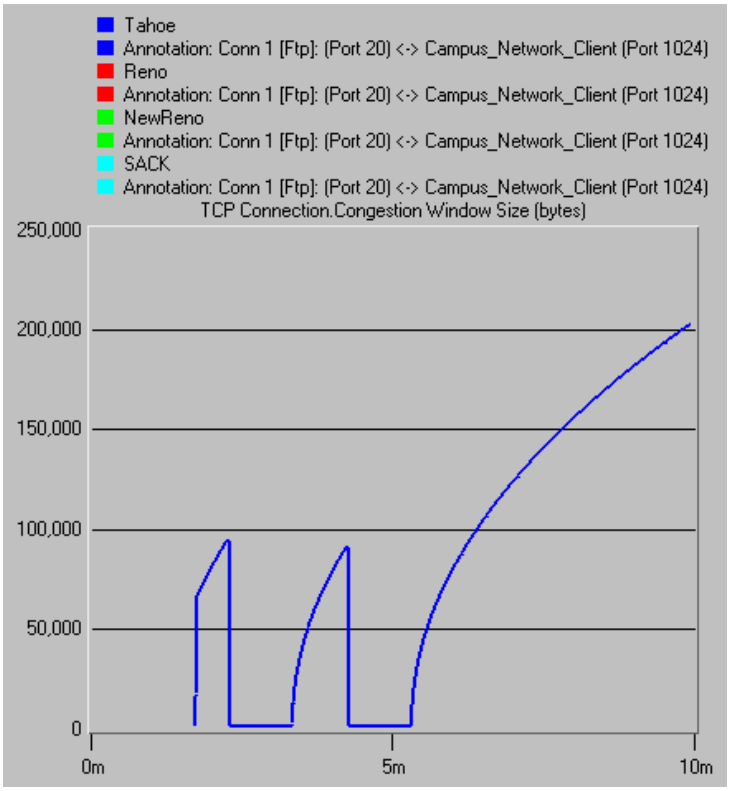

Fig.16. CWND in wireless topology

As shown in figure 16, the congestion window size for all TCP congestion control algorithms (Tahoe, Reno, NewReno, and SACK) overlap each other and it verifies that all algorithm reach to a same result in case of signal attenuations in wireless network.

Thus, all TCP congestion control algorithms which mentioned here are designed to work in wired networks efficiently. In other respects, they do not perform satisfactory in a wireless network due to signal attenuations, obstacles, fading, and multi path in wireless network environment.

Conclusion

In this project, we have introduced the congestion mechanisms and the congestion control algorithms in TCP. We have desired to understand and compare the behaviors of congestion control algorithms in terms of congestion window size and sent segment sequence number in wired networks when packet drops occur. Overall results show that NewReno and SACK outperforms other algorithms in terms of congestion window size and a file download time. Reno has a significant problem when multiple packets drop in a window.

Since fading, obstacles, signal attenuation, and multi-path cause packet drops in wireless network, we have simulated a wireless topology in Opnet Modeler and we have used a trajectory for a wireless mobile client to cause packet drops when the mobile client goes beyond a certain radius. As a result, all congestion control algorithms perform same in terms of congestion window size when signal attenuations happen in wireless networks.

References

1) K. Fall and S. Floyd, "Simulation based comparisons of Tahoe, Reno and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, July 1996, pp. 5-21.
2) M. N. Akhtar, M. A. O. Barry and H. S. Al-Raweshidy "Modified Tahoe TCP for wireless networks using OPNET simulator," *Proc of the London Communications Symposium (LCS2003),* London, Sept 2003.
3) W. Stevens, "TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms,". Network Working Group, RFC2001, Jan 1997.
4) M. Omueti and Lj. Trajkovic, "M-TCP+: using disconnection feedback to improve performance of TCP in wired/wireless networks," *Proc. SPECTS 2007, San Diego*, CA, USA, July 2007, pp. 443-450.
5) A. S. Tanenbaum, "Computer Networks," The Transport Layer, 4rd Edition.