King Fahd University of Petroleum & Minerals

Department of Computer Engineering

COE 541

Local and Metropolitan Area Networks

Term 091

Project Progress Report # 3[Final]

TCP/IP window and slow-start study over wireless channels

Submitted to: Professor Mayez Al-Mouhamed

Submitted By	Student ID
1. Mohammed Abdul Khadir Khan Asif	200702570
2. Adeniye Suli	200803940

26th Jan, 2009

Table of Contents

List of Figures

Project Problem Statement

TCP/IP window and slow-start study over wireless channels, Assume two nodes are communicating using TCP/IP. The TCP initializes a transmission window and starts sending sequenced segments. The window size increases as the session progresses increasing the utilization of the link. When congestion is occurring (lost segments or their acknowledgments) the TCP reduces the window size to reduce outgoing traffic. This leads to lower utilization of the link. After some time of no data loss, TCP increases the window size again to increase the link utilization and so on. For wireless links, packets may be lost due to errors and not necessarily due to congestion and buffer overflow. Write a simulation code that emulates this process. Determine the actual parameters used in current TCP/IP based transport. For a given arrival process of segments at the source nodes, and a given error model on the wireless link, find the average throughput of the wireless link and the average segment delay.

1. Introduction

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite (the other being Internet Protocol, or IP), so the entire suite is commonly referred to as *TCP/IP*. Whereas IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two end systems, for example a Web browser and a Web server. In particular, TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer. Among its other management tasks, TCP controls segment size, flow control, the rate at which data is exchanged, and network traffic congestion [6].

Congestion control is the problem of managing network traffic or a network state where the total demand for resources such as bandwidth among the competing users exceeds the available capacity. It is a core infrastructural problem stemming from the packet switched and statistically multiplexed nature of the Internet and has an impact on the Internet stability and manageability.

Although TCP is the most common transport protocol used in the Internet for years, it has been shown that its congestion control algorithm lacks the ability to adapt to the wireless environments. TCP is primarily designed for wired networks, where data is seldom lost or corrupted due to link errors and queue overflow in routers is the predominant reason for the packet loss. In a wireless network, however, packet losses will occur more often due to high Bit Error Rates (BERs) than due to congestion. When using TCP over wireless networks, it considers each packet loss as a sign of congestion and invokes congestion control measures at the source. This results in severe performance degradation [5].

It is highly undesirable for a protocol to react to random losses the same way as it reacts to congestion indications. TCP should react to congestion rather than packet losses. Due to the characteristics of the air interface, wireless links could introduce sporadic packet losses due to burst of packet losses. A loss of a single packet often has a little effect on network applications, but multiple packet losses can have a significant effect. TCP's inability

to distinguish a loss due to congestion from a random loss can lead to serious performance degradation.

Moreover, TCP can yield low throughput in highly mobile environments due to hand-offs, which may introduce temporal disconnections, buffer losses and increased latency. Shadowing and fading of the radio signal may also cause the destination to be temporarily unavailable, which causes the TCP either to time out or to stop the transmission. The lack of mechanism to notify the TCP of this effect introduces extra delay and increases the application response time considerably.

In summary, TCP is very sensitive to packet losses and requires further improvements to better adapt to the wireless environments. It should be able to distinguish wireless related losses from the congestion related losses and be fine tuned to utilize the available network resources efficiently

1.1. TCP Congestion Control Scheme:

Slow Start & Congestion Avoidance: When a TCP connection begins, the value of CongWin is typically initialised to 1 MSS, resulting in an initially sending rate of roughly MSS/RTT. Since the bandwidth available to the connection may be much larger than MSS/RTT, a TCP sender increases its sending rate exponentially by doubling its value of CongWin every RTT. The TCP sender continues to increase its sending rate exponentially until there is a loss event, at which time CongWin is cut in half and then grows linearly (Congestion Avoidance phase). The initial phase of the connection is called Slow Start. During the initial phase, the TCP sender begins by transmitting at a slow rate (hence the term Slow Start) but increases its rate exponentially until it reaches a value called Threshold which determines the window size at which slow start will end and Congestion Avoidance will begin. The variable Threshold is initially set to a large value (65K, in practice) so that it has no initial effect. Whenever a loss event occurs, the value of the Threshold is set to one half of the current value of CongWin.

A loss event can be due to a TimeOut or due to the receipt of three (3) duplicate acknowledgements at the sender. A receipt of 3 duplicate acknowledgements at the sender for a segment of transmission indicates that that the sent segment is not received.

After a TimeOut event, a TCP sender enters a Slow Start phase. While in slow-start, it increases the value of CongWin exponentially fast until CongWin reaches Threshold. When CongWin reaches Threshold, TCP, again, enters the congestion avoidance phase, during which CongWin increases linearly.

In short, a TCP Sender additively increases its rate when it perceives that the end-to-end path is congestion free and multiplicatively decreases its rate when it detects (via a loss event) that the path is congested. For this reason, TCP congestion control is often referred to as an additive-increase, multiplicative-decrease (AIMD) algorithm.

Slow Start and Congestion avoidance algorithms thus provide features like flow control in error recovery situations and adaptation to network conditions. However, when an error occurs, these algorithms respond relatively slowly in recovering the network back to its original throughput. Fast Retransmit and Fast Recovery are two other algorithms which help the network get back to its original throughput. These two algorithms are briefly explained as follows.

Fast Retransmit: In case of a segment loss, an immediate acknowledgement is sent. This is also called a duplicate acknowledgement. Fast Retransmit algorithm requires that if 3 acknowledgments are received before the retransmission timeout, the sender then has to retransmit the lost segment immediately. From this point in time until an acknowledgement for new data arrives, every duplicate acknowledgement triggers a new data transmission. Thereafter, the *cwnd* is set back to its initial value and *Slow Start* is initiated.

Fast Recovery: This algorithm is based on the notion that if duplicate acknowledgements are being received, then the network is not fully congested. Thus there is no need to abruptly reduce the flow and initiating a *Slow Start.* Hence, on receiving 3 duplicate acknowledgements, the *cwnd* is set to half of its previous size and *Congestion Avoidance*, instead of *Slow Start*, is performed. The combinations of Fast Retransmit and Fast Recovery give rise to the different flavours of TCP namely Reno, Tahoe and SACK.

1.2. TCP Flavours:

Tahoe TCP: Fast Recovery algorithm does not allow the *cwnd* to open exponentially. Instead, it applies the much slower Congestion Avoidance mechanism. Due to this, problems arise in Reno TCP when multiple segments are lost. To avoid this problem, Tahoe TCP does not apply Fast Recovery. When this flavour is used, Fast Retransmit is the only algorithm applied. Figure 1-1 shows the behaviour of TCP Tahoe.

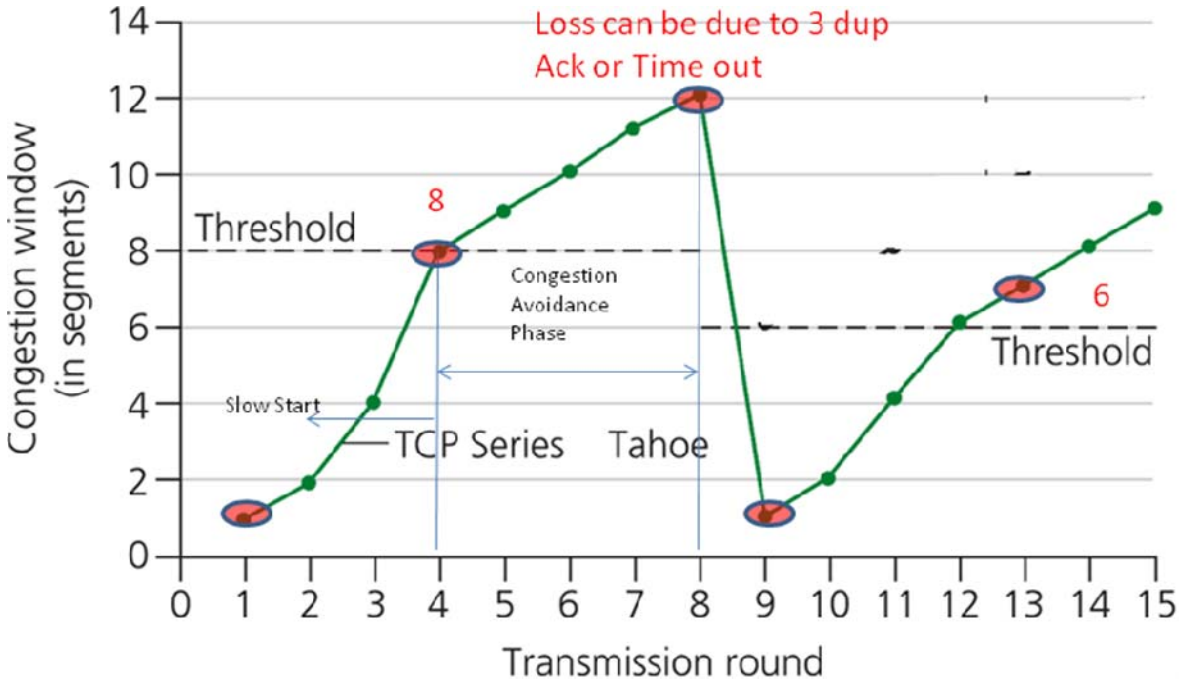

Figure 1-1: Additive-increase, multiplicative-decrease congestion control (TCP Tahoe)

Reno TCP: The flavour that uses both Fast Retransmit and Fast Recovery together is called Reno TCP. Reno performs well until only one segment is lost, but in a wireless scenario, there will be multiple segment drops in a handover and also in the unreliable media. In case of multiple segment losses, the *cwnd* closes to half of its previous size for each segment drop. Furthermore, the recovery is further delayed because only the Congestion Avoidance algorithm is functioning without the added benefit of Slow Start. Figure 1-2 shows the behaviour of TCP Reno.

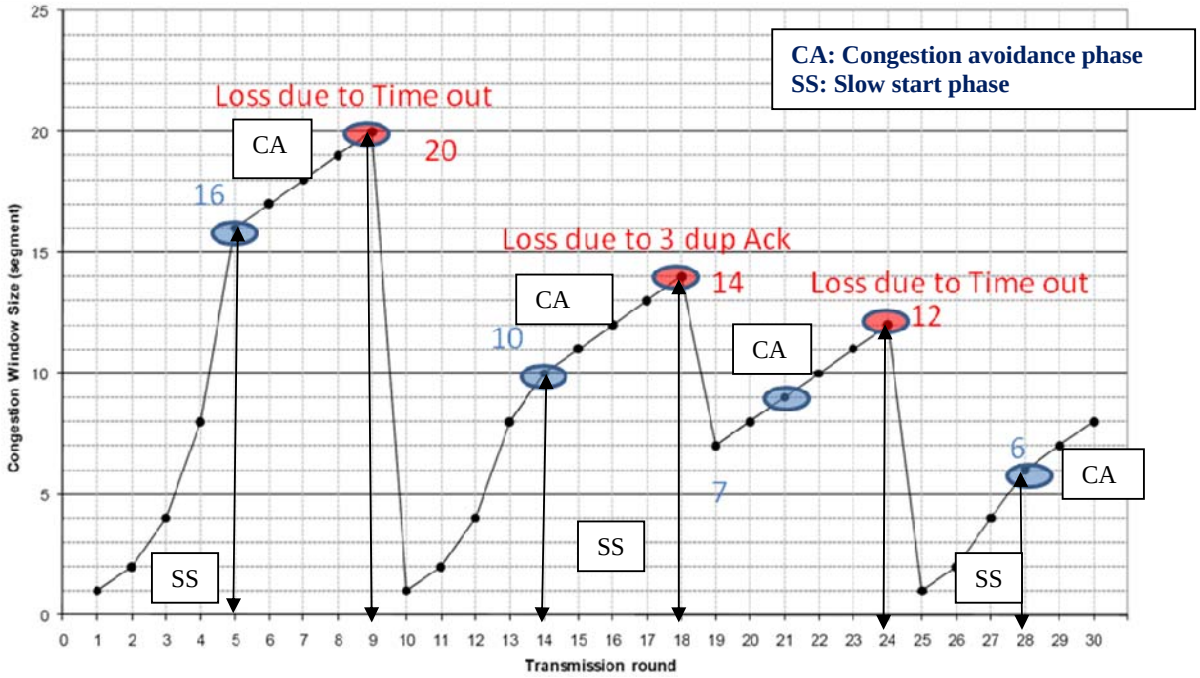

Figure 1-2: Additive-increase, multiplicative-decrease congestion control (TCP Reno)

New-RENO TCP: New RENO is a slight modification over TCP-RENO. It is able to detect multiple packet losses and thus is much more efficient than RENO in the event of multiple packet losses. Like Reno, New-Reno also enters into fast-retransmit when it receives multiple duplicate packets, however it differs from RENO in that it doesn't exit fast-recovery until all the data which was outstanding at the time it entered fast-recovery is acknowledged. Thus it overcomes the problem faced by Reno of reducing the CongWin multiples times.

Summary of different TCP flavours' features:

TCP algorithms \ TCP flavours	*TCP Tahoe*	*TCP Reno*	*TCP New Reno*	*TCP SACK Reno*
Slow start	✓	✓	✓	✓
Congestion avoidance	✓	✓	✓	✓
Fast Retransmit	✓	✓	✓	✓
Fast Recovery	N/A	N/A	✓	✓
Handling of Multiple Packet Loss	N/A	N/A	✓	✓
Selective Acknowledgement	N/A*	N/A*	N/A*	✓

*Cumulative acknowledgement

2. Literature Review

Wireless links are inherently lossy [3] and in addition to random losses, they suffer from long period of fading as well. TCP has no mechanism to differentiate these losses from congestion and therefore treats all losses as congestive by reducing its transmission window and in effect reducing the throughput of the connection.

After a packet loss, TCP reduces its transfer data speed by switching to slow start followed by congestion avoidance phase. While this scheme is helpful in dealing with congestion in wired network since the total traffic offered is reduced and thus, congestion is alleviated, the scheme may reduce the TCP throughput significantly in wireless network and will lead to unacceptable performance especially if underlying network is capable of supporting high data rates.

Varshney [1] designed an algorithm termed "Selective Slow Start" to differentiate between segment losses due to network congestion and that due to the handoffs based on pattern of losses (timeout). His idea is that TCP will only initiate slow start only if the number of lost TCP segments in the last S attempts exceeds a limit. To achieve this, he relied on the reasoning that:

i) Handoffs will occur less frequently say, one, in 60 seconds and will last for a small interval and therefore will cause a packet loss infrequently and the loss is likely to involve consecutive TCP segments, and that

ii) If Congestion occurs it will last longer than the handoff duration and is likely to cause loss of several TCP segments in a given interval of time.

To implement this idea as an algorithm, TCP needs to keep track of segment losses and the time of their loss. The result is that, after a timeout, TCP will check if the number of segments loss in the last S attempts including the current one, exceed a limit. If it does, then TCP will assume that these losses have been caused by network congestion and will initiate a slow start. Otherwise, it will continue to transmit at its current speed. Varshney's simulation results of the SSS algorithm therefore improves transport layer throughput significantly in wireless environment.

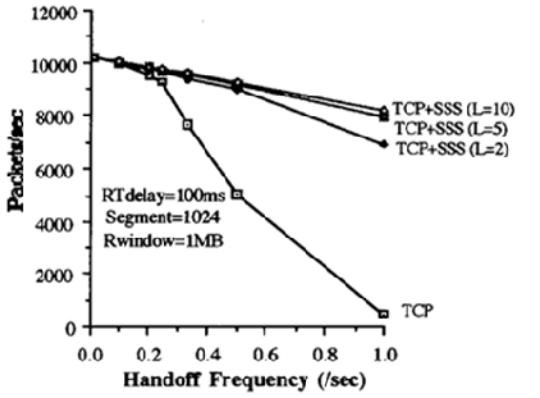

Figure 2-1: hroughput Vs Handoff Frequency

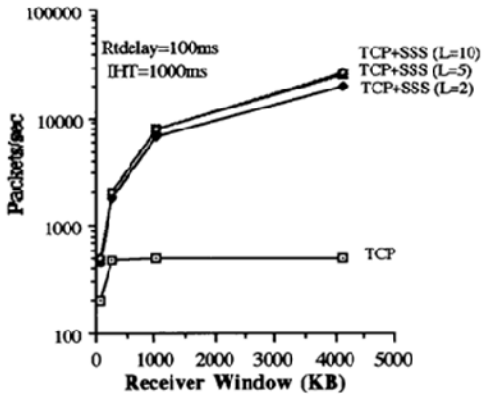

Figure 2-2: CP Throughput vs Receiver Window

$$LIMIT = K * C * T_h / Seg$$

Throughput in TCP-SSS is better than that of Normal TCP

K-safety factor to compensate for peak values
C= avg # of bits transmitted per sec
Th=Avg handoff duration in sec
Seg=Avg TCP segments size

Figure 2-1, 2-2: Throughput Vs Handoff Frequency and TCP throughput Vs Receiver Window

Explanation of Figures 2-1, 2-2: It can be seen from the Fig (1) that when handoffs are very infrequent, TCP and TCP without SSS perform nearly the same. However, as the frequency of handoff increases, the TCP throughput drops much faster than that of TCP with SSS and at very high handoff frequency i.e. 1 per second, representative of pico cells and fast moving wireless hosts, TCP throughput is close to zero.

In their work on "Techniques to Improve TCP over Wireless links", Aaron and Tsao [2] discussed the end-to-end schemes and link layer schemes that have been proposed and simulated by various authors and gave an analytical reviews (based on effectivity and practicality) on their proposals. Their analysis described the response of unmodified TCP to a packet loss and a discussion on the effect of TCP algorithms to the performance of a wireless link. In addition, they also discussed different proposals and issues regarding end-to-end techniques and presents and optimum solution.

Parca and Garcia-Luna-Aceves [3] presented a simple method in which a protocol (TCP Santa Cruz) is able to differentiate random losses from losses due to congestion,

thereby avoiding the rate-halving approach taken by standard TCP whenever any loss is detected. TCP Santa Cruz is a new implementation of TCP that detects not only the initial stages of congestion in the network, but also identifies the direction of congestion, i.e. determines whether congestion is happening in the forward or reverse path of the connection. They relied on the method of relative delays to determine congestion and to monitor the increasing and decreasing congestion in the network. The result of their simulation showed that TCP Santa Cruz is able to maintain a high throughput over wireless link for a range of error rates because it does not reduce its sending rates when losses are determined to be random losses on the wireless link.

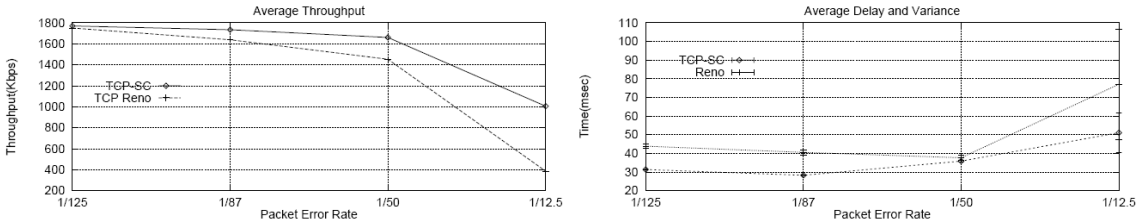

Figure 2-3: TCP Santa Cruz and Reno for different error rates (a) Throughput (b) Delay and variance

Figure 2-3: TCP Santa Cruz and Reno for different error rates (a) Throughput (b) delay and variance

Explanation of the Figure 2-3: The graph labelled (a) shows that TCP Santa Cruz is able to maintain high throughput over the wireless link for a range of error rates because it does not reduce its sending rate when the losses are determined to be random losses on the wireless link. TCP Reno, on the other hand, can make no such distinction and as a result shows reduced performance, even when link error rates are low. It can also be seen from graph (b) that packets transmitted with TCP Santa Cruz experience a lower end-to-end delay and delay variation from source to receiver because the protocol keeps the bottleneck queue at a minimum level and does not create the oscillations in bottleneck queue length that is typical of the TCP Reno transmission pattern.

In an attempt to study congestion in wireless networks, Jardosh et al [4] used link layer information collected from an operational, large-scale and heavily utilised IEEE 802.11b wireless network deployed at an IETF meeting. In their work, they motivated the use of “channel busy time” as a direct method of channel utilisation and showed how channel utilisation along with network throughput and goodput can be used to define “high

congested", "moderately congested", and uncongested network states. Their result correlates congestion and its effect on link performance.

The throughput of the channel for a one second interval is the sum of the total number of bits of all recorded frames transmitted over the wireless channel during a one second interval. The goodput of the channel is the total number of recorded bits of all the control and successfully acknowledged data frames transmitted over the wireless channel during a one second interval.

Figure 2-4 indicates that as the channel utilization increases from 30% to 84%, the average throughput of the channel increases to 4.9 Mbps and the average goodput increases to 4.4 Mbps. The average throughput at 84% channel utilization is closest to the achievable theoretical maximum throughput. Since the computation of throughput includes all the transmitted frames, the calculated throughput value for each utilization percentage is higher than the corresponding value of goodput.

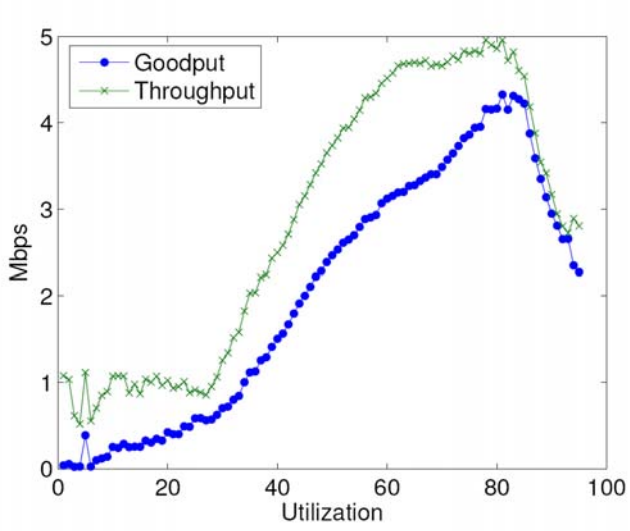

Figure 2-4: Channel throughput and goodput per second calculated at the corresponding channel utilization

TCP over Wireless Networks:

Effect of High BER: Bit errors cause TCP data segments or ACKs loss. When ACKs do not return within RTO, the sender retransmits the segment, exponentially backs off its RTO (up to 64 seconds), and slow start again. Repeated errors result in low throughput.

Effect of Disconnections: Disconnections can be caused by a handoff, physical obstacles, or call blocking. These disconnections result in lost data segments and lost ACKs, and greatly reduce the efficiency of the connections.

Effect of Frequent Disconnections: Small cell sizes result in small cell latencies and cause frequent disconnections as a user roams. It may result in a serial timeouts. When the mobile is reconnected, no data is successfully transmitted for minutes!

Effect of Route Recomputations: When an old route is no longer available, the network layer at the sender attempts to find a new route to destination. It is possible that discover a new route is longer than RTO and invokes congestion control.

Effect of Network Partitions: The ad hoc network may periodically get partitioned for several seconds at a time. If the sender and the receiver lie in different partitions, the data packages and ACKs will get dropped and result in congestion control.

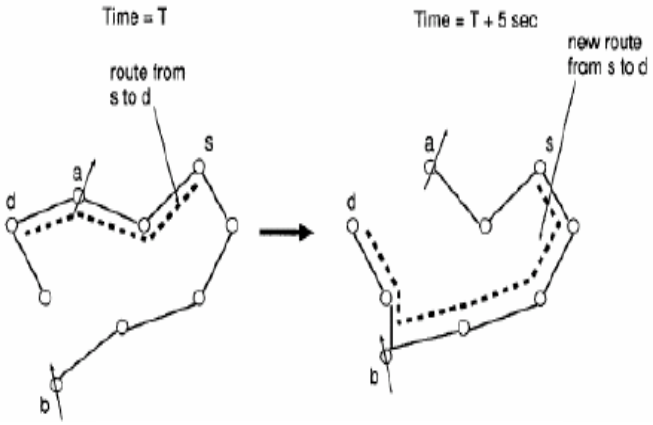

Figure 2-5: Effect of Network Partitions

3. Simulation Results

For the simulation purpose, we use OPNET Modeler 14.0 [7]. The below network model is tested for different TCP flavours. The network has a mobile client downloading a file from a server using the File Transfer Protocol (FTP). The mobile client connects to the Wireless Router, which serves as the Access Point. The network is configured in such a way that the mobile node has the freedom to move around the Access point in a defined trajectory.

As the mobile node moves far away from the Access Point, transmission errors are generated due disconnection from the access point. Different error probabilities are created in the network as a result of the mobile node moving in different trajectories. Error rates of

0.5%, 1.0%, 1.5% & 2.0% are simulated for the mobile node when it is downloading a 1 MB, 2 MB, 5MB and 20MB file sizes from the server. The file download response time for each error rate is recorded for each flavour of TCP (Tahoe, Reno, and New Reno). Each simulation is carried out for 6 runs, and the average of the 6 runs is considered. Response time (sec) is measured from the time an application data packet is sent from the source TCP layer to the time it is completely received by the TCP layer in the destination node.

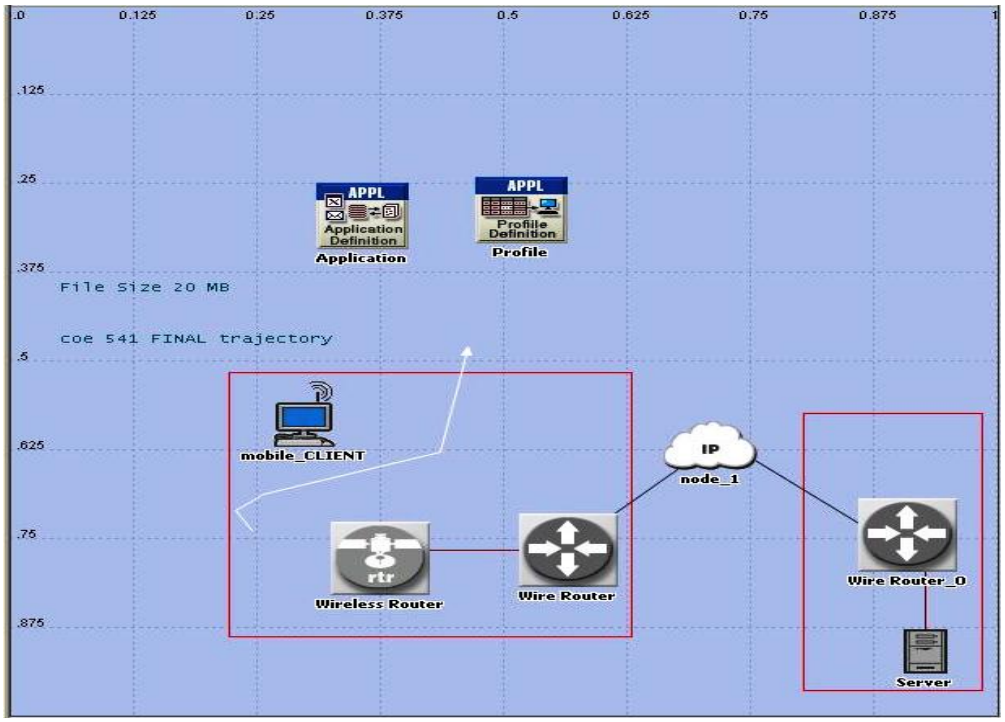

Figure 3-1: Network Model

Figure 3-2 shows the behaviour of *TCP Tahoe* with 0.5% Packet Drop. It employs three transmission phases: Slow Start, Congestion Avoidance and Fast Retransmit. Here, the congestion window goes frequently to 1MSS (1460 bytes) after the detection of packet loss. The sending rate reduces either due to the reception of triple duplicate acknowledgement or timeout. The file download response time is 38sec.

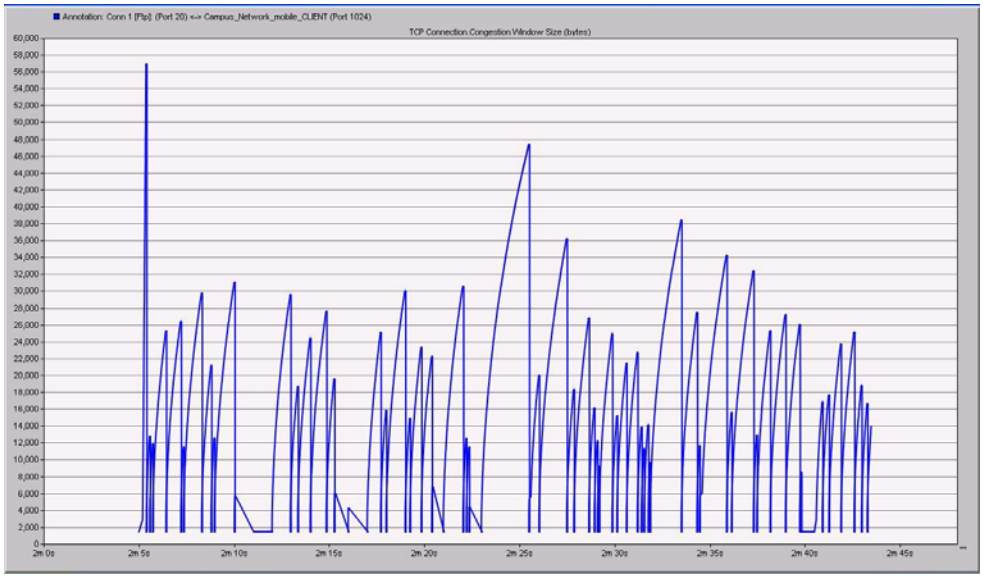

Figure 3-2: Tahoe with 0.5% Drop, File Size: 20MB

Figure 3-3 shows the behaviour of *TCP Reno* with 0.5% Packet Drop. It employs transmission phases – Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery. Here, the congestion window goes to 1MSS (1460 bytes) if the detection of packet loss is due to timeout. However, the congestion window reduces to half of its current value if the loss is detected due to triple duplicate acknowledgement. The File download response time is 40 Sec.

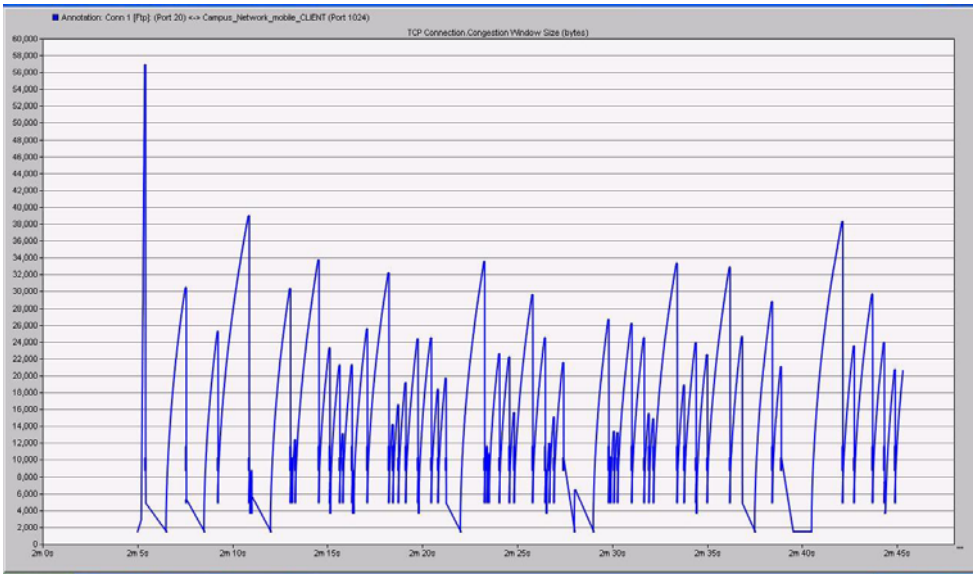

Figure 3-3: TCP Reno with 0.5% Drop, File Size: 20MB

Figure 3-4 shows the behaviour of *TCP NewReno* with 0.5% Packet Drop. The New-Reno, being an improvement of TCP Reno, is better suited to handle the problem of multiple packet losses than that of TCP Reno. The File download response time is 33 Sec.

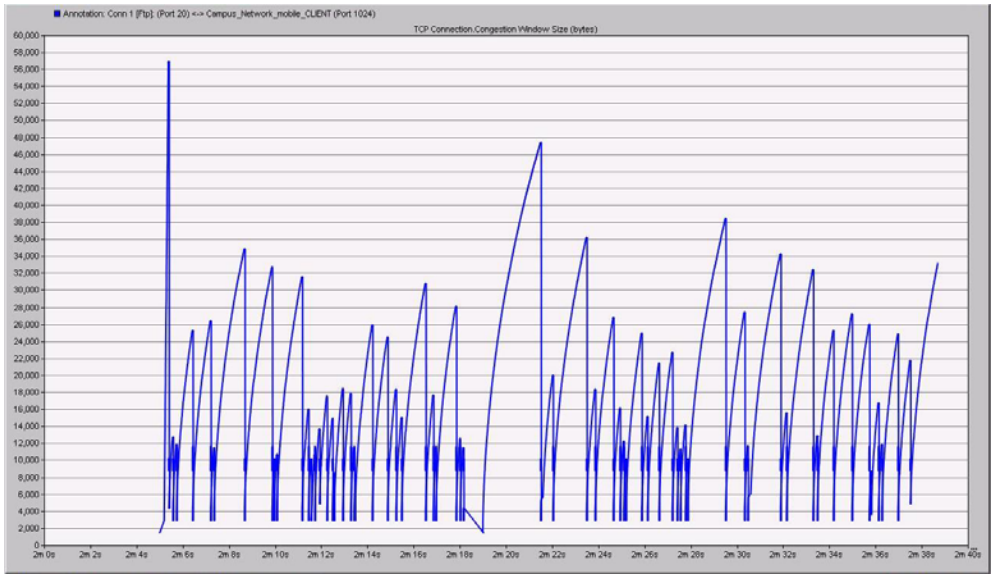

Figure 3-4: TCP NewReno with 0.5% Drop, File Size: 20MB

Figure 3-5 shows the file download response time for the 1 MB FTP file for different error rates.

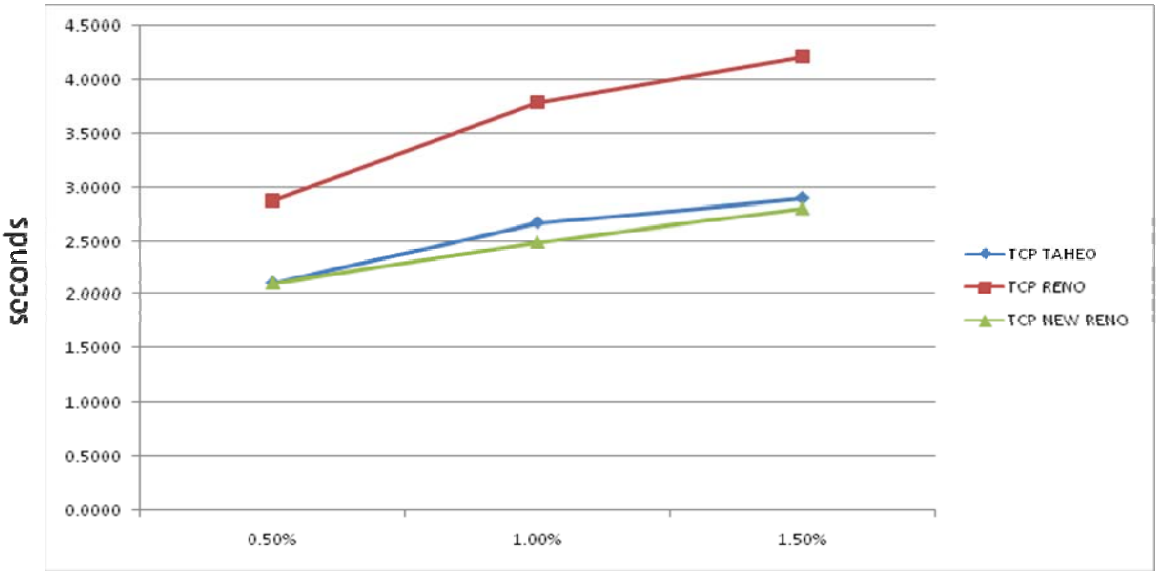

Figure 3-5: File download response time (in seconds) v/s different percentage error drop, File Size: 1 MB

From the graph

Figure 3-6 shows the file download response time for the 2 MB FTP file for different error rates.

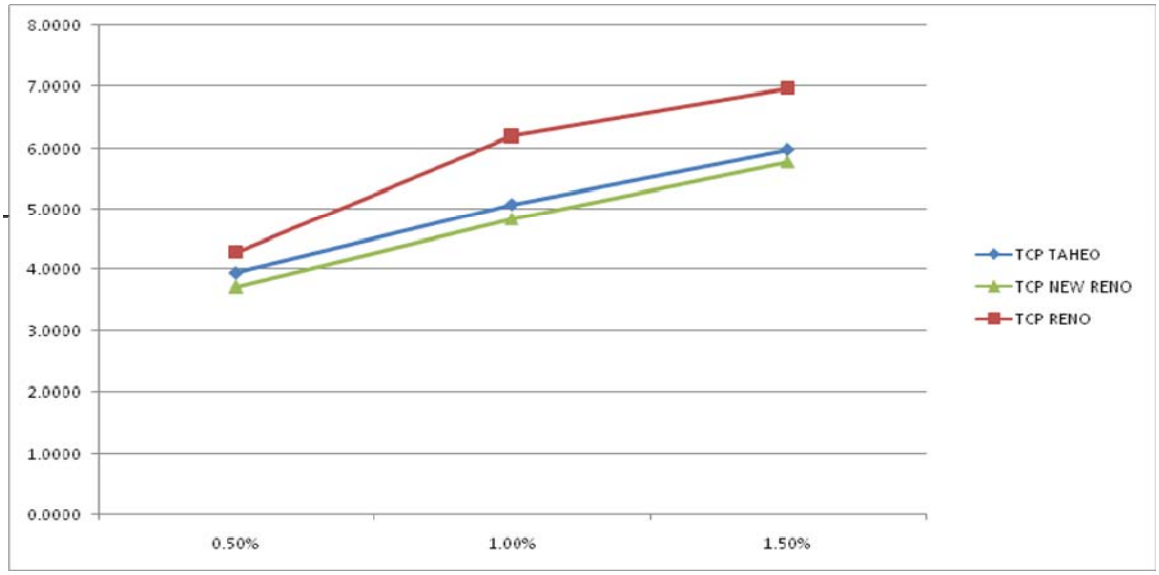

Figure 3-6: File download response time (in seconds) v/s different percentage error drop, File Size: 2 MB

Figure 3-7 shows the file download response time for the 5 MB FTP file for different error rates.

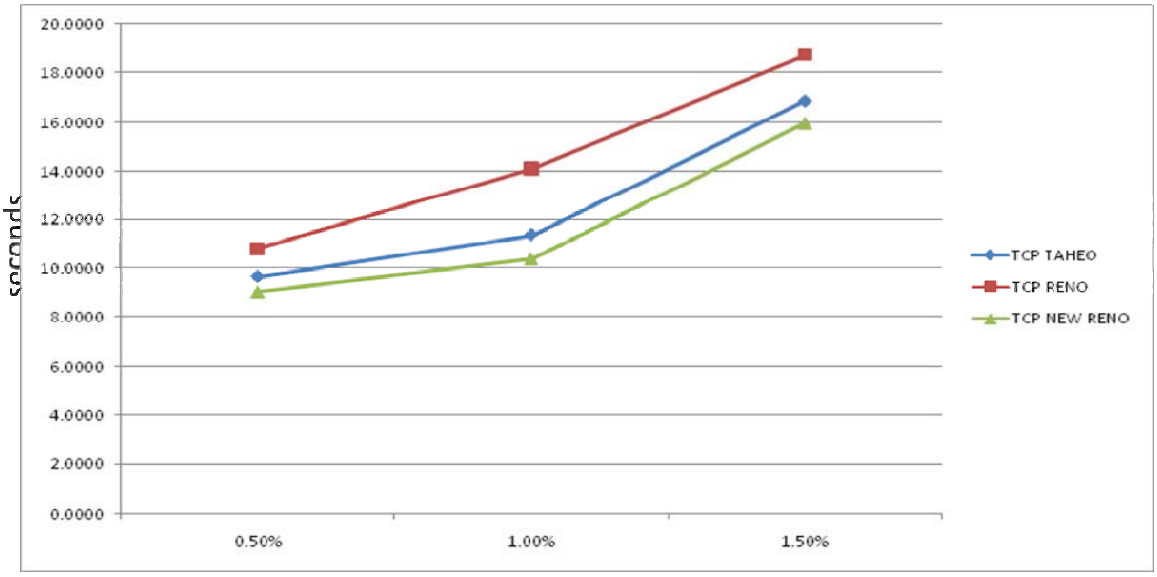

Figure 3-7: File download response time (in seconds) v/s different percentage error drop, File Size: 5 MB

Figure 3-8 shows the file download response time for the 20 MB FTP file for different error rates.

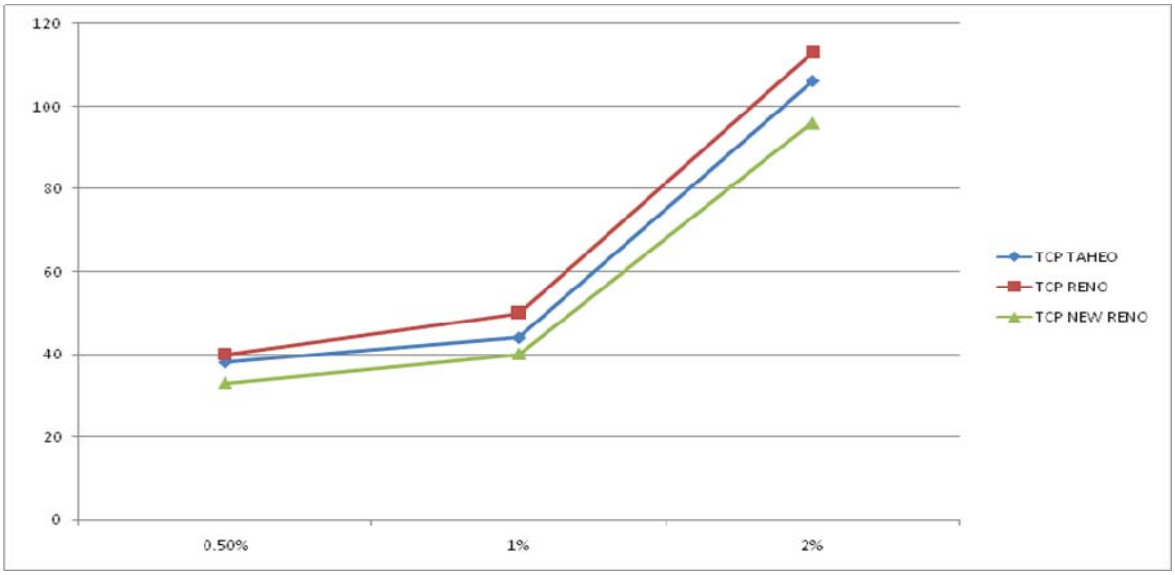

Figure 3-8: File download response time (in seconds) v/s different percentage error drop, File Size: 20 MB

Analysis of results for figures 3-5, 3-6, 3-7, 3-8: From all the plots, we observe that as error rate increases for all the TCP flavours, the file download response time also increases. This extra overhead is due to retransmission of packets that are not acknowledged by the receiver since they are in error. When TCP Reno is used, the download response time is more. This is because after the reception of three duplicate acknowledgements, the congestion window goes to half of the current congestion window and TCP enters a congestion avoidance phase in which it increases the congestion window by 1 MSS every transmission round and increases linearly. On the other hand, TCP Tahoe goes to 1 MSS, after the reception of three duplicate acknowledgement or TimeOut, but increases the Congestion Window exponentially. After the reception of three duplicate acknowledgements, TCP NewReno maintains the current congestion till the last segment is acknowledged before dropping to half. This gives the TCP NewReno performance enhancement over other TCP flavours in wireless network.

4. Conclusion

In this report, we studied the behaviour of TCP in wireless environment. This is done by simulating different TCP algorithms in OPNET 14.0 Modeler. The performances of various TCP flavours are compared in wireless environment with different error rates. FTP File Download response time increases as percentage of packet drop increases. Network is tested with different percentage of packet drop and with different file size. In all the simulation

runs, we observed TCP NewReno maintain larger Congestion Window size than the remaining protocols. From the above results we concluded that TCP New Reno performs better compared to other flavors of TCP.

5. References

[1]. Upkar Vashney, "Selective Slow Start: A Simple Algorithm for Improving TCP Performance in Wireless ATM Environment", IEEE Computer, 1997.

[2]. Anne Aaron and Susan Tsao, "Techniques to Improve TCP Over Wireless Links", Final Report, EE 359, Stanford University, December 2000.

[3]. Christiana Parsa and J.J. Garcia-Luna-Aceves, "Differentiating Congestion vs Random Losses: A Method for Improving TCP Performance over Wireless Links", IEEE WCNC 2000.

[4]. J. Jardosh, K. Ramachandran, K. Almeroth and E. Belding, "Understanding Congestion in IEEE 802.11b Wireless Networks", In ACM/USENIX International Measurement Conference, Berkeley, CA, October 2005.

[5]. A. Jayananthan, "TCP Performance Enhancement over Wireless Networks", Phd Thesis, University of Canterbury, New Zealand, 2007.

[6]. J. F. Kurose, K. W. Ross, Computer Networking: A top down approach featuring the Internet, 3rd edition, Addison Wealey, 2005,

[7]. www.opnet.com