International Journal of Information Engineering and Applications
2018; 1(1): 11-17
http://www.aascit.org/journal/information

AASCIT American Association for Science and Technology

TCP Flavors Simulation Evaluations over Noisy Environment

Elsadig Gamaleldeen Elsadig Karar

Department of Computer Sciences, Faculty of Applied Sciences & Computer, Omdurman Ahlia University, Khartoum, Sudan

Email address

elsadigg@gmail.com

Citation

Elsadig Gamaleldeen Elsadig Karar. TCP Flavors Simulation Evaluations over Noisy Environment. *International Journal of Information Engineering and Applications.* Vol. 1, No. 1, 2018, pp. 11-17.

Received: January 14, 2018; **Accepted**: January 29, 2018; **Published**: February 12, 2018

Abstract: The Transmission Control Protocol (TCP) Flavors was simulated using OPNET simulation tool in a noisy wireless environment to observer TCP behavior and select the suitable protocol to be used. The simulation was done in a client server network and get the output from the packet analyzers. The simulation is done in the noisy environment with a configuration of a low bandwidth starting from 128 Kbps up to 1 MB, burst traffic and a high delay in RTT time to give a really environment and the result was that the TCP New Reno is the best flavor that can be used in the noisy environment.

Keywords: TCP Flavors, Simulation, Noisy Environment

1. Introduction

In this Paper, the simulation of the TCP protocol was implemented and discussed, there is real demand to setup simulation environment to monitor TCP behavior. The analysis and design are mainly done by the OPNET simulation tool.

2. Method

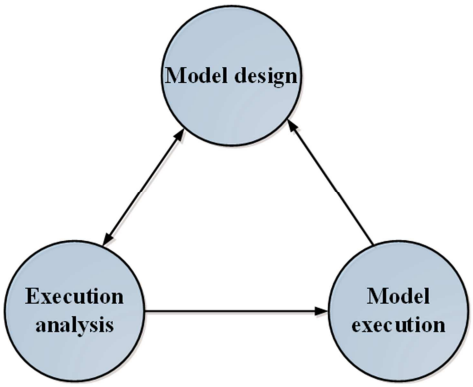

Figure 1. *Model design life cycle.*

The simulation is done through computers by modeling a design for the systems and theoretical running systems, simulations are executed through normal computers that are available for the collection of the relative informational results and the model design life cycle illustrated in Figure 1. Learning by doing, is the main concept about systems in study which modeling is required and operating them [1].

2.1. OPNET Architecture

OPNET provides a comprehensive development environment for modeling and performance evaluation of communication networks and distributed systems. The package consists of a number of tools, each one focusing on particular aspects of the modeling task [2]. These tools fall into three major categories that correspond to the three phases of modeling and simulation projects: Specification, Data Collection and Simulation Analysis.

These phases are necessarily performed in sequence. They generally form a cycle, with a return to specification following analysis. Specification is actually divided into two parts: initial specification and re-specification, data collection and simulation cycle, as illustrated in Figure 2.

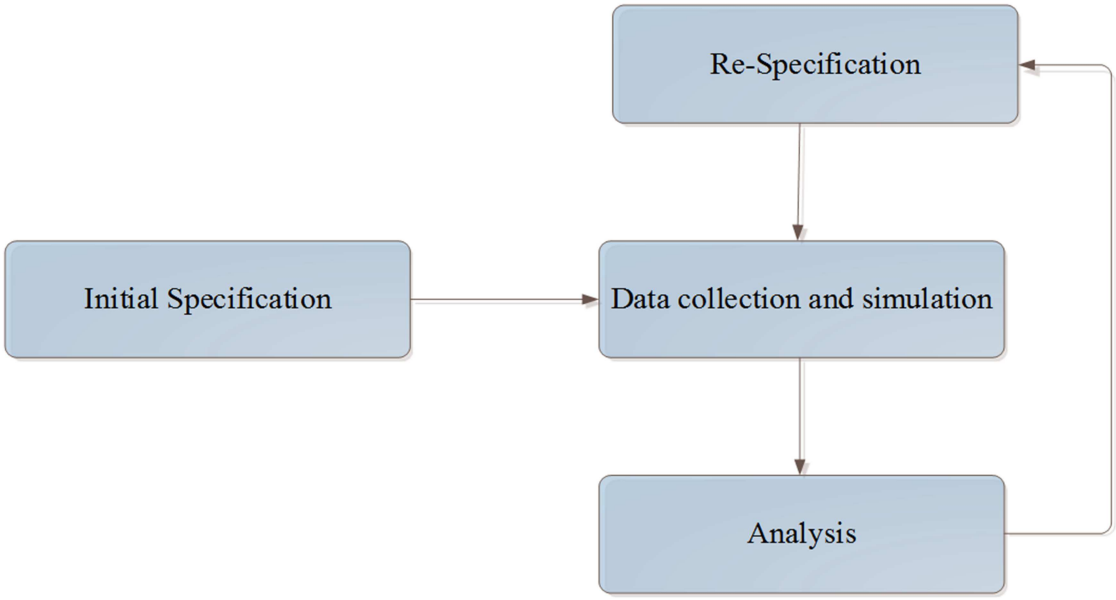

Figure 2. *OPNET Project cycle simulation.*

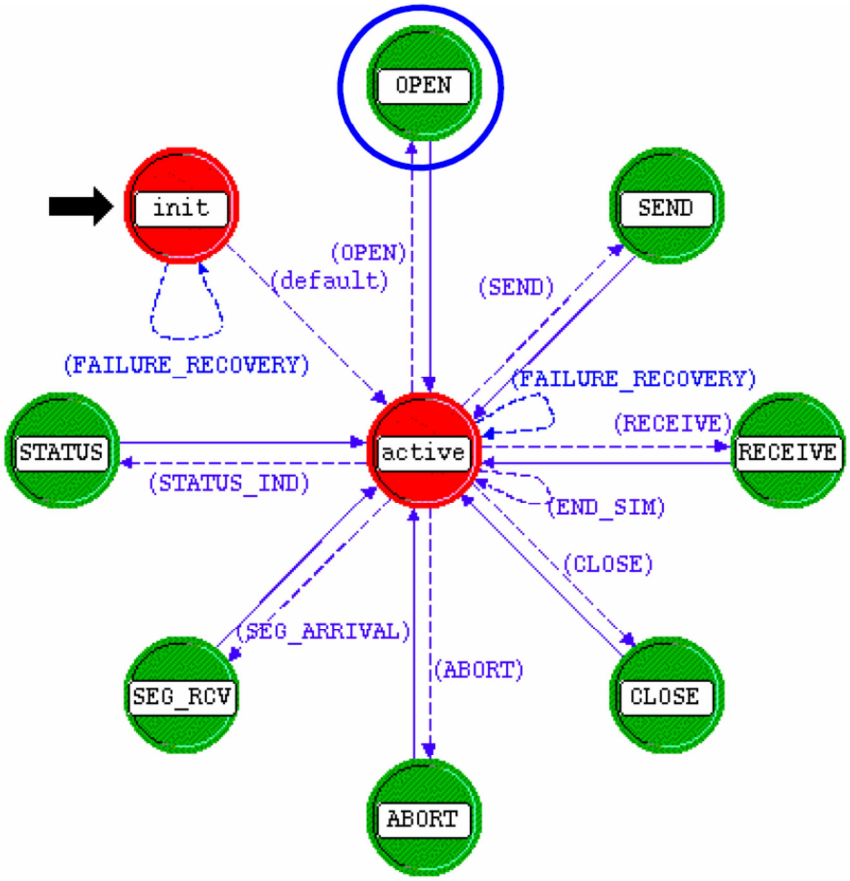

Figure 3. *OPNET TCP process model.*

2.2. TCP Implementation in OPNET

The design that had been made was begin with process model for the TCP initial process as Figure 3, node editor and the network editor after that it follows the procedures until the result. The study had cover TCP behavior and the modeled of the TCP flavors it had examined:

a. Segment retransmission process (the order in which segments are retransmitted).
b. Fast recovery vs. expiration of timeout timer.
c. Congestion window after drops

The simulation had also compared the result of the original version of TCP with the modified one.

2.3. The Simulation Network Level

The scenario which had been designed to implement the TCP protocol was consist of two nodes client and server with two Packet analyzers one from the link server to client and the other from client to server Figure 4 explain the network level in the simulation scenario.

The Packet analyzers which drop designated packets to observe TCP behavior and traffic can intercept and log traffic passing over the network or part of a network. As data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications that configure in the simulation.

The output from the packet analyzers that generated during the simulation phase can be used as main data to be study after the compilation process in the reports.

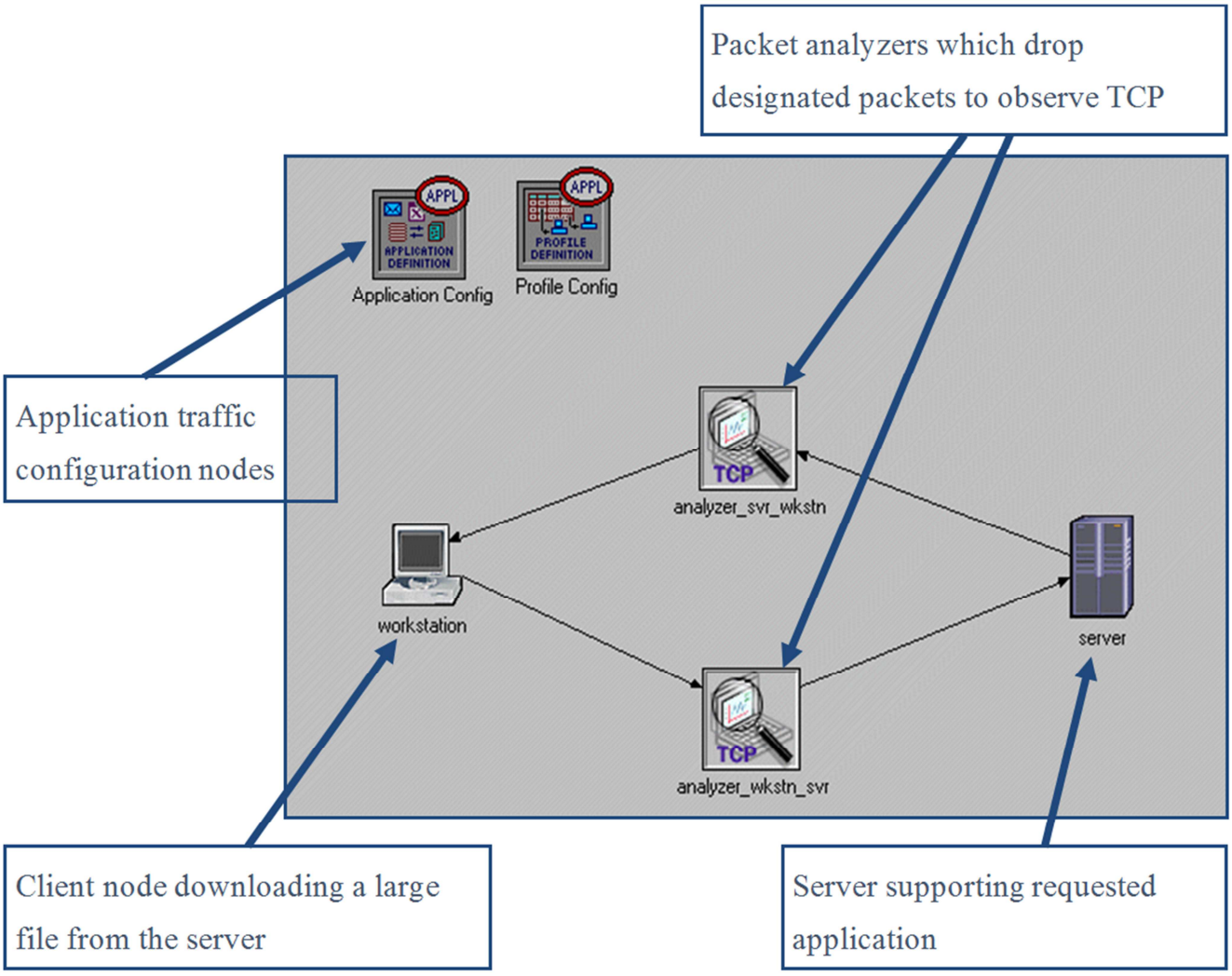

Figure 4. *Network level in the simulation scenario.*

2.4. Simulation Configuration

The simulation had been configuring to run requests for the FTP from the client node to the server and TCP detects a packet loss when:

a. Retransmission timer expires.
b. Duplicate acknowledgement is received for the connection.

The TCP flavors differ in how they react to packet loss. While all TCP implementations reset the congestion window after retransmission timeout expiration to one maximum segment size (MSS), they may proceed differently after duplicate ACKs are received. The missing segment is always resent immediately, but transmission of new or unacknowledged data depends on the selected flavor.

The design had configured to compare the following flavors:

a. Tahoe – fast retransmit followed by slow start.
b. Reno – fast retransmit followed by fast recovery.
c. New Reno – similar to Reno, but does not have congestion window and multiple times during recovery process.
d. Selective acknowledgement (SACK) – selective retransmission based on received selective acknowledgements.

3. Result

In the simulation, the Receive Buffer is set to 65,535 bytes (64 kbps). Since this value is greater than the size of transmitted file, the size of the TCP receive buffer will never prevent data from being sent.

The Maximum ACK Delay is 0.001 sec. This will cause ACKs to be sent almost immediately after receiving a segment. The TCP Packet Analyzer node had been used to selectively drop segments and to introduce additional network delay to the segment. Segments to be dropped are identified by their order. In this scenario, there are three segments (23, 27, and 28) that are dropped by the *analyzer_server_workstation* node Figure 5.

The simulation is done in the noisy environment with a configuration of a low bandwidth starting from 128 Kbps up to 1 Mbps, burst traffic and a high delay in RTT time to give a really environment to system.

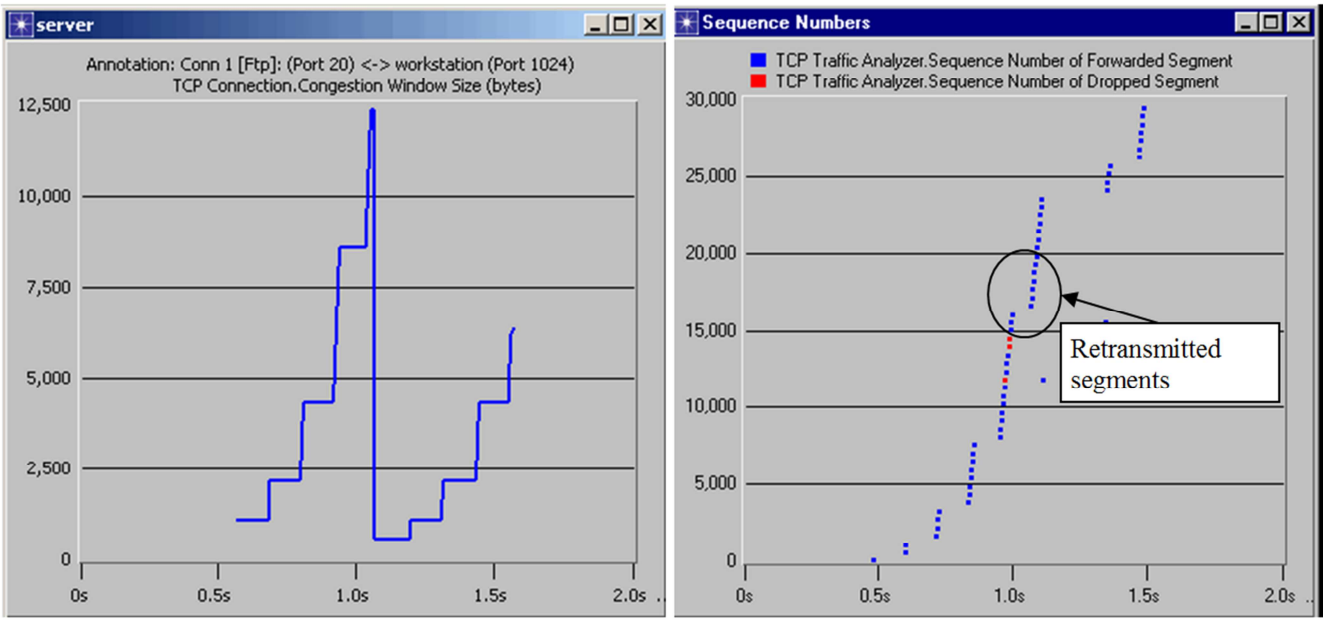

Figure 5. *Sequence Numbers and ACK Numbers.*

TCP Tahoe simulation test results

TCP Tahoe supports fast retransmission and resets the congestion window to one MSS after segment retransmission. It is the simplest one out of the four variants. It doesn't have fast recovery [3]. At congestion avoidance phase, it treats the triple duplicate ACKs same as timeout. When timeout or triple duplicate ACKs is received, it will perform fast retransmit, reduce congestion window to 1, and enters slow-start phase. In Opnet simulation the Tahoe behave in Fast retransmit, no fast recovery and packet drops introduced long delays in application response time due to the resetting of the congestion window to one after fast retransmission.

TCP New Reno simulation test result

The TCP New Reno on Hosts had been configuring by enabling New Reno instead of fast recovery on both hosts by changing "Fast Recovery" to "New Reno" configuration on all the nods.

TCP New Reno differs from TCP Tahoe at congestion avoidance. When triple duplicate ACKs are received, it will halve the congestion window, perform a fast retransmit, and enters fast recovery [4]. If a timeout event occurs as explained in Figure 6, it will enter slow-start, same as TCP Tahoe. TCP New Reno is effective to recover from a single packet loss, but it still suffers from performance problems when multiple packets are dropped from a window of data.

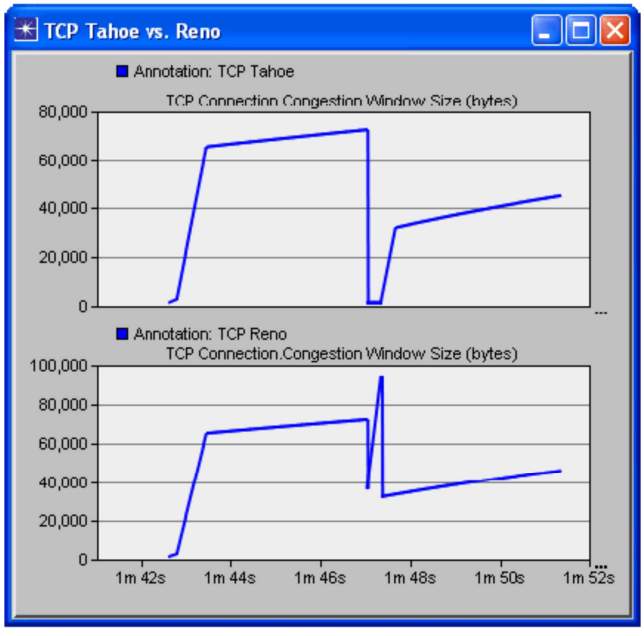

Figure 6. *Comparison between TCP Tahoe and TCP New Reno.*

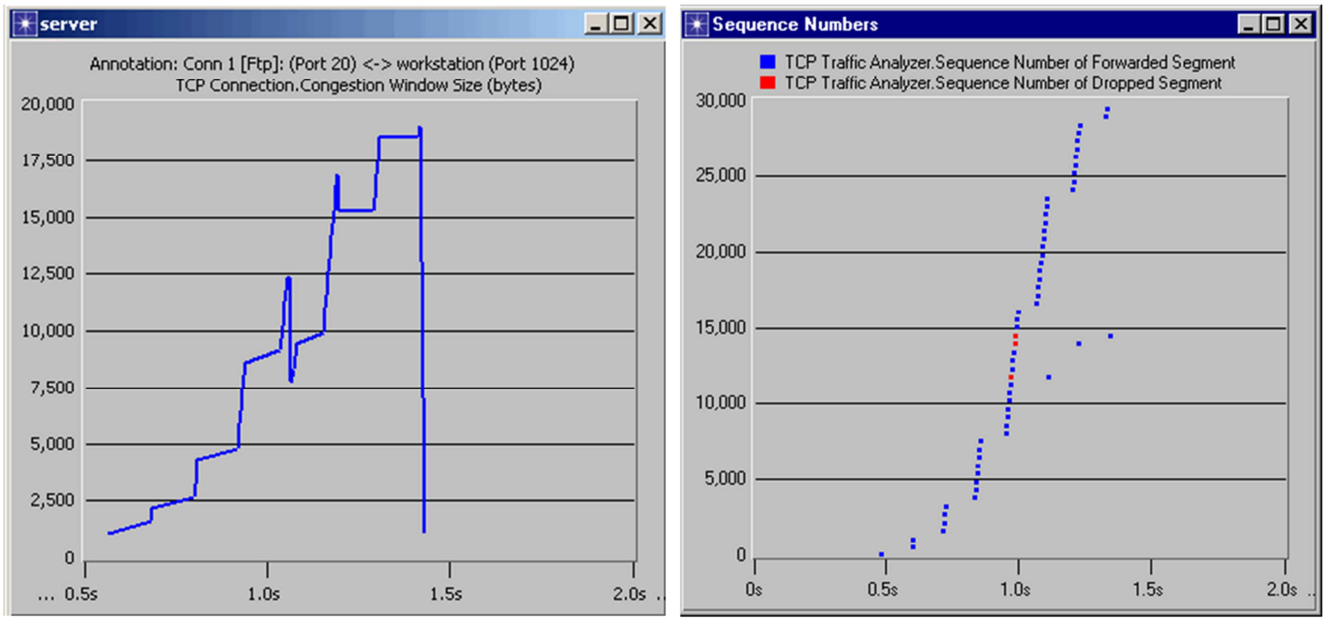

Figure 7. *TCP New Reno on Hosts Result.*

When run the simulation for New Reno the following points represent the result:

a. Congestion window is halved after fast retransmission and then increased by one MSS for each received ACK.
b. Fast recovery ends when ACK for retransmitted data is received.
c. Congestion window is then set to half of pre-fast-retransmission value.
d. Assumes that multiple segments from the same window of data are not likely to be lost.
e. Accurate most of the time.
f. Because of this assumption, multiple segment losses can produce application response times that are worse than Tahoe TCP Figure 7.

The New Reno is suitable to be used in simulation for the TCP noisy environment because it's clear after the simulation is that:

a. Only halve the congestion window once for one window of data
b. Do not end fast recovery when ACK for retransmitted segment is received, but rather end it only after ACK for the last ACK segment at the time the fast-retransmitted segment is ACK Figure 8.
c. Very minor fix to Reno algorithm.

But there is a Problem with TCP Reno if multiple packets are dropped from the same window of data, congestion window will be halved multiple times [5]. This can lead to slower response times than if not using Fast Recovery at all. For that the New Reno will be used to avoid this problem.

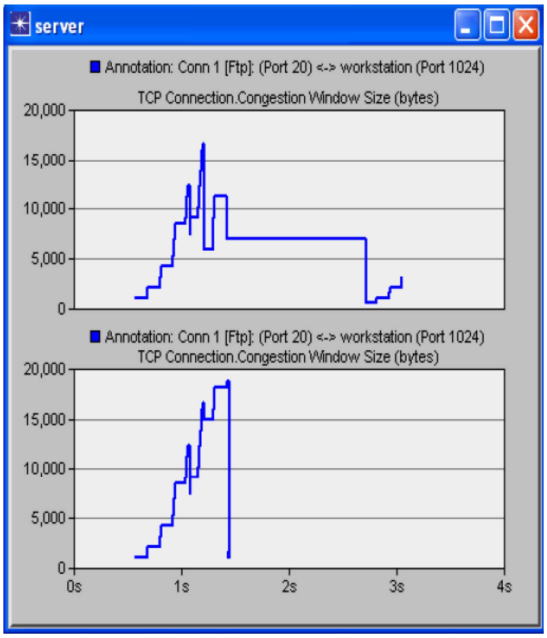

Figure 8. *TCP Reno vs TCP New Reno on server site.*

TCP SACK simulation test result

When configuring TCP SACK on Hosts and run the Simulation kernels the collect results are generated by duplicated the New Reno scenario and name it "SACK".

SACK TCP adds a number of SACK blocks in TCP packet [6], where each SACK block acknowledges a non-contiguous set of data has been received. The main difference between SACK TCP and Reno TCP implementations is in the behavior when multiple packets are dropped from one window of data Figure 9. SACK sender maintains the information which packets is missed at receiver and only retransmits these packets. When all the outstanding packets at the start of fast recovery are acknowledged, SACK exits fast

recovery and enters congestion avoidance.

After an expiration of the round-trip time out timer in all the other TCP flavors it clear that sender retransmits all un ACK data, even though some might have been successfully delivered and additional information carried in TCP header allows receiver to ACK data selectively report blocks of received data [7]. The TCP SACK is not suitable to be used in the noisy environment and that due to the following points:

a. Sender can inter missing data and retransmit "selectively".
b. Implementations are typically Reno-based.
c. In order for a connection to support SACK, both ends must support it.

This is due that sack is a window based [8] that uses the options field of TCP to precisely inform the sender which segments were not received [9]. This enables the sender to retransmit only those segments that are lost, thereby not to waste network bandwidth [10].

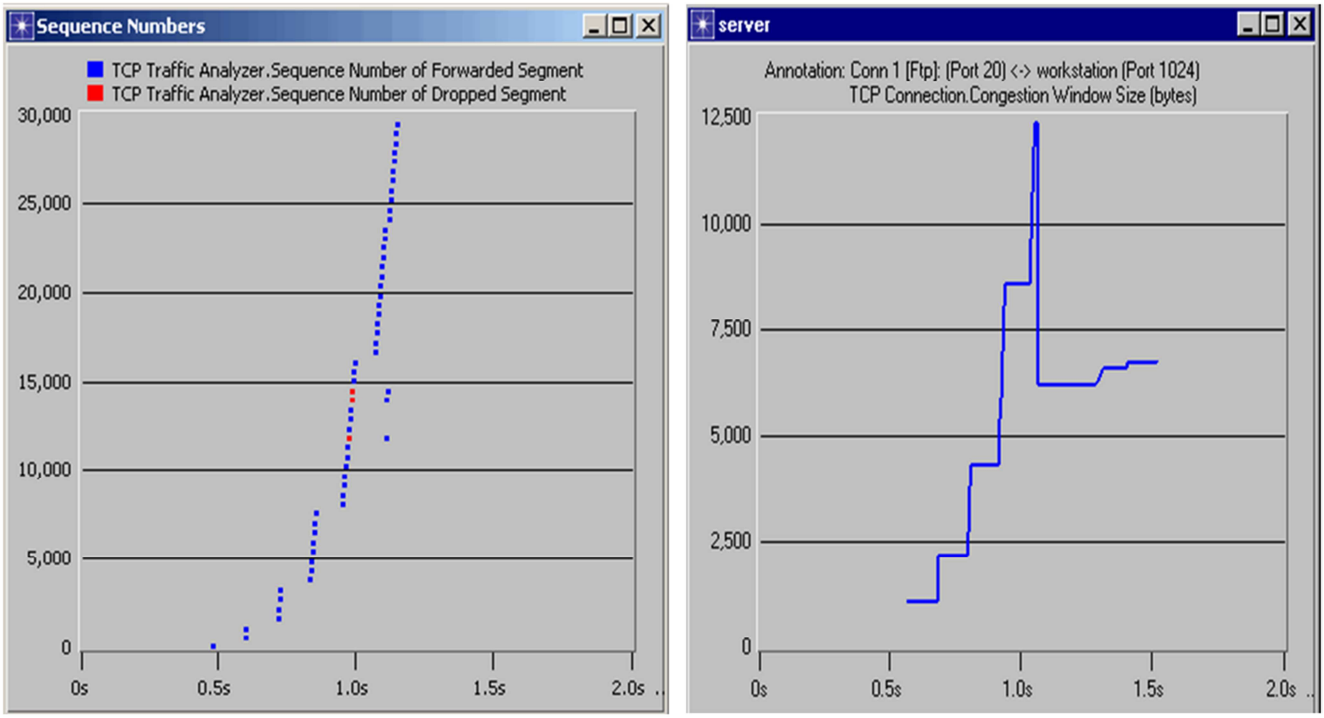

Figure 9. *TCP sack on Hosts result.*

4. Discussion

To make a comparison between the flavors many concepts had been consider and they are:

a. Observe TCP behavior after multiple packet loss
b. Monitor congestion window
c. Collect application response time for different flavors

Due to the server heavy traffic, the simulation result was taken from the workstation node.

As seen in Figure 10 the simulation for the flavors in the noisy environment and High RTT value the comparison result is:

a. ECN offers the best performance followed by SACK and New Reno.
b. Reno has the worst response time for this case.
c. In contrast, for a one segment drop, Reno performs better.

From the above comparison, the best TCP flavors to be used to test the estimation of TCP noisy environment is TCP New Reno.

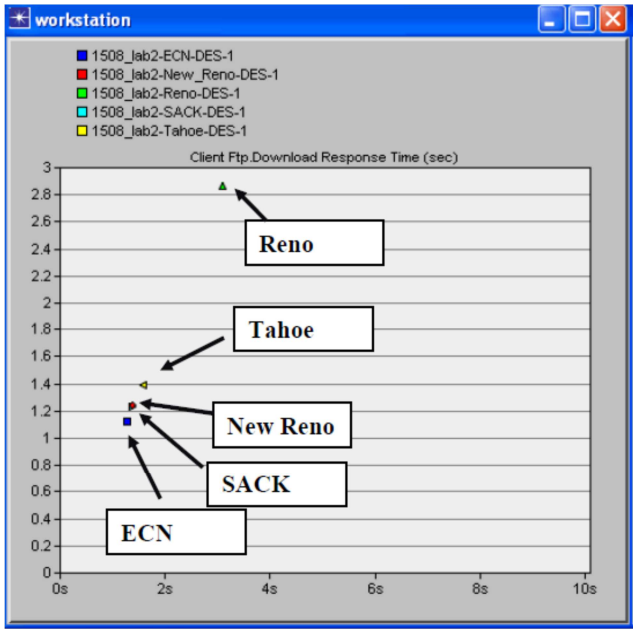

Figure 10. *TCP flavors comparison.*

5. Conclusion

In a noisy environment and when there is a low bandwidth TCP flavor the New Reno is suitable to be used in simulation for the TCP noisy environment because it's clear after the simulation is only halve the congestion window once for one window of data and do not end fast recovery when ACK for retransmitted segment is received.

References

[1] Caini and R. Firrincieli, 2004. "Packet Spreading Techniques to Avoid Bursty Traffic in Long RTT TCP Connections", in Proc. IEEE VTC Spring, Italy.

[2] Omogbohun Omueti, Modupe, 2007. TCP-ADaLR: TCP WITH ADAPTIVE DELAY AND LOSS RESPONSE FOR BROADBAND GEO SATELLITE NETWORKS. Master of applied science. USA: Simon Fraser University.

[3] N. Dukkipati and N. McKeown, 2006. Why Flow-Completion Time is the Right Metric for Congestion Control. ACM SIGCOMM Computer Communication Review, vol. 36, no. 1.

[4] A. Vishwanath, V. Sivaraman, and D. Ostry, 2009. "How Poisson is TCP Traffic at Short Time-Scales in a Small Buffer Core Network?" In Proc. IEEE Advanced Networks and Telecommunication Systems (ANTS), India.

[5] Mr. R. D. Mehta, Dr. C. H. Vithalani, Dr. N. N. Jani, 2010. "Enrichment of 'SACK' TCP performance by delaying fast recovery", International Journal of Advanced Engineering Technology E-ISSN 0976-3945.

[6] Constantinos Dovrolis, 2005. "Passive and Active Network Measurement-New Methods for Passive Estimation of TCP Round-Trip Times": 6th International Workshop, PAM 2005, Boston, MA, USA.

[7] A Razdan, ANandan, R Wang, MY Sanadidi, M Gerla, 2002. "Enhancing TCP Performance in Networks with Small Buffers, Proceedings", IEEE/ICCCN' 2002, Miami, Florida, Nov. 2002.

[8] Russ White, 2014. The Art of Network Architecture: Business-Driven Design (Networking Technology). 1st Edition. Cisco Press.

[9] IP Micro-Mobility Home Page, 2016. IP Micro-Mobility Home Page. [ONLINE] Available at: http://www.comet.columbia.edu/micromobility.

[10] Steel Central for Performance Management and Control | Riverbed application and network performance management solutions | Riverbed. 2017. Steel Central for Performance Management and Control | Riverbed application and network performance management solutions | Riverbed. [ONLINE] Available at: http://www.opnet.com.