Modeling and Simulation of Out-of-Order Impact in TCP Protocol

Seifedine Kadry and Ali Edan Al-Issa

***Abstract*—The Internet provides a powerful, cost effective, reliable, and survivable communication over the world for different services like e-commerce, finance, education, military, and so on. The Internet has been rapidly increased; it has been growing at unprecedented rates. As the Internet become huge, its network performance becomes subjected to a lot of problems that decrease the communication performance like congestion, interference, reordering and so on.**

In this paper, we focus on measuring the impact of packets and ACKs reordering on the performance of TCP variants namely TCP Reno, TCP New Reno, TCP SACK, and TCP Vegas, we also analyze the performance of these protocols when their duplicate acknowledgement threshold (Dupthresh) values are increased in order to prevent false fast retransmissions and unnecessary Congestion Window Size (CWND) reductions.

***Index Terms*—TCP, out-of-order, NS-2, TCP Reno, TCP new Reno, TCP SACK, TCP Vegas.**

I. Introduction

The Internet communication is managed by set of protocols grouped by several layers, these protocols and layers construct the communications framework model or suite, TCP/IP suite one of the most common of all network protocol suites. TCP/IP is a hierarchical protocol made up of interactive layers (Fig. 1).

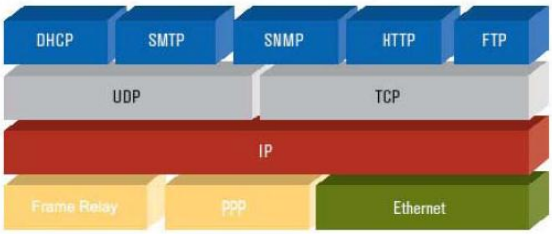

Fig. 1. TCP/IP layers.

Application layer are placed at the top of TCP / IP stack [1], it include many protocols, such as (FTP, HTTP, SMTP and so on) for application communication. The transport layer follows the application layer. TCP/IP makes available two distinct transport layer protocols to the application layer: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). In our thesis, we depend Transmission Control Protocol (TCP) which is the most commonly used protocol in the Internet, because it is used for so many application protocols such as HTTP, POP, SMTP, etc. to transfer data reliably between the source and destination. TCP provides reliable connection between the sender and the receiver by using sequence number for each packet and ACKs that confirm the delivery of the packet to the destination, TCP uses several techniques and algorithms to regulate the transmission rate at which the sender must send such as congestion control and flow control and etc.

The recent researches [2], [3] appear that the occurrence of packet reordering on the network is not pathological behavior. Bennett *et al.* [4] had intended that reordering is highly prevalent on many links. The results of their study [4] indicate that the probability of a session, running through the US MAE-East exchange, experiencing packet reordering was over 90%. Intuitively, Bennett et al. referenced that the reason for the large proportion of flows experiencing reordering was the presence of parallelism on the routes taken by the packets flowing through the network. A test was conducted by Stanford Linear Accelerator Center (SLAC), the test included 256 sites and data packets were sent from SLAC to all these sites. It was found that roughly 25% of the sites exhibit reordering of packets (http://www-iepm.slac.stanford.edu/monitoring/reorder/2000).

In this paper, we focus on measuring the impact of packets and ACKs reordering on the performance of TCP variants TCP Reno, TCP New Reno, TCP SACK, and TCP Vegas, we also analyze the performance of these protocols when their Dupthresh values are increased in order to prevent false fast retransmissions and unnecessary CWND reductions.

II. Transmission Control Protocol (TCP)

The Internet provides a convenient and cost-effective communication platform for electronic commerce, education, and entertainment. The success of the Internet arises from its capabilities to support survivable, robust, and reliable end-to-end data transfer services a lot of applications running over a set of end-systems. The Internet is originated from the Advanced Research Projects Agency Network (ARPANET) designed to support survivable military communications. Currently, Transmission Control Protocol (TCP) [5] is the most popular transport layer protocol that provides process-to-process, full duplex, in-order delivery, connection oriented, flow control, Error control, and reliable data transfer in the Internet. TCP is one of the two original components of the TCP/IP suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as TCP/IP. TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). TCP [6] provides the service of exchanging data directly between two network host's

Manuscript received November 12, 2014; revised June 14, 2015.

S. Kadry is with American University of the Middle East, Kuwait (e-mail: skadry@gmail.com).

A. E. Al-Issa was with Arts Sciences and Technology University, Lebanon (e-mail: eng.aliedan@yahoo.com).

processes by adding port number for each process in the port field of the TCP header, whereas IP handles addressing and routing data across one or more networks. In particular, TCP allows the sending process to send the data as a stream of byte and allow the receiver to obtain the data as stream of byte, so TCP creates an imaginary environment in which the process seem to be connected by an imaginary "tube" that carries their bytes across the internet. The processing (write and read data) on the both direction may not be at the same rate, so TCP need buffer for storage on both direction (sender and receiver). The Network layer, which work as services provider for Transport layer, need to send data in packet format not as stream of byte, so TCP groups number of bytes together into a packet or segment. A TCP connection provides a full-duplex service: If there is a TCP connection between Process A on one host and Process B on other host, then application layer data can flow from Process A to Process B at the same time as application layer data flows from Process B to Process A. TCP is connection oriented protocol that means when application process at one of the host wants to send or receive data from another application process at the other host, TCP must establish virtual connection (not physical) and make" handshaking" with each other then data are exchanged in both side, after sending data the connection is terminated.

The TCP [7] must recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted (segment), and requiring an acknowledgment (ACK) from the receiving TCP. Segment sequence number is the byte-stream number of the first byte in the segment, that's mean when a process in Host A wants to send a stream of data to a process in Host B over a TCP connection, the TCP in Host A will implicitly number each byte in the data stream, if the data stream consists of a file consisting of 100,000 bytes, the MSS (maximum segment size) is 1,000 bytes, and that the first byte of the data stream is numbered 0, the second segment gets assigned sequence number 1,000, the third segment gets assigned sequence number 2,000, and so on. Each sequence number is inserted in the sequence number field in the header of the appropriate TCP segment. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. When a destination receives a data segment, it acknowledges the receipt of the segment by issuing an acknowledgement (ACK) with the next expected data octet number (segment). The time elapsed between when a data segment is sent and when an ACK for the segment is received is known as the round-trip time (RTT) of the communication between the source and the destination, which is the sum of the propagation, transmission, queuing, and processing delays at each hop of the communication, and the time taken to process a received segment and generate an ACK for the segment at the destination.

III. Congestion Control

An important issue in the Internet is congestion. Congestion in a network may occurs if the load on the network (the number of packet sent to the network) is greater than the capacity of the network (the number of packets a network can handle). Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Congestion in the network occurs because routers and switches have queues (buffers that hold the packets before and after processing). The packet is put in the appropriate output queue and waits its turn to be sent. These queues are finite, so it is possible for more packets to arrive at a router than the router can buffer.

There are two broad approaches of congestion control that are taken in practice, these types are:

- End-to-end congestion control. In an end-to-end approach to congestion control the network layer provides no explicit support to the transport layer for congestion control purposes. The presence of congestion in the network must be inferred by the end systems based only on observed network behavior (for example, packet loss and delay). TCP segment loss (as indicated by a timeout or a triple duplicate acknowledgment) is taken as an indication of network congestion and TCP decreases its window size accordingly.
- Network-assisted congestion control. With network-assisted congestion control network-layer components (routers) provide explicit feedback to the sender regarding the congestion state in the network. This feedback may be as simple as a single bit indicating congestion at a link.

The TCP congestion-control mechanism operating at the sender keeps track of three variables per connection:

- The Congestion Window (CWND) is a sliding window, which constraint on the rate at which a TCP sender can send traffic into the network, before receiving an Acknowledgement.
- The Receiver's Advertised Window (RWND) is flow control from the Receiving TCP, indicating a window size of data the receiver is willing to accept.
- The Slow Start Threshold (Ssthresh) is a value used to decide whether the Sending TCP is transmitting packets using the Slow Start or Congestion Avoidance algorithm, and if the CWND variable should be adjusted.
- Specifically, the amount of unacknowledged data at a sender may not exceed the minimum of CWND and RWND, that is:

```
LastByteSent - LastByteAcked < min {cwnd, rwnd}
```

There are several methods and techniques for improving the performance of TCP by limiting the sending rate of data packet by limiting the amount of unacknowledged segments allowed to be sent, and the size of the congestion window; we will discuss these techniques just after we summarize the TCP algorithms.

IV. TCP Algorithms [8]

A. Slow Start

Slow start is conducted in the beginning of every TCP connection and its main purpose is to find the maximum

available bandwidth at which it can send data without ca using the network to be congested.

B. Congestion Avoidance

If the receiver window is large enough, the slow start mechanism will start discarding packets.

C. Fast Retransmission

If an out-of-order segment is received TCP generates a so called duplicate acknowledgment. This duplicate acknowledgment is sent immediately from the receiver to the sender indicating that a segment arrived out-of-order, and which segment that was supposed to be received.

D. Fast Recovery

After fast retransmit is conducted, congestion avoidance and not slow start is performed. This behavior is called Fast Recovery.

V. TCP Variants

A. TCP Tahoe

Tahoe refers to the TCP congestion control algorithm. TCP Tahoe is based on a principle of 'conservation of packets', i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well.

B. TCP Reno

This Reno [9] retains the basic principle of Tahoe, such as slow starts and the coarse grain re-transmit timer. However it adds some intelligence over it so that lost packets are detected earlier and the pipeline is not emptied every time a packet is lost.

C. TCP New Reno

New RENO [10], [11] is a slight modification over TCP-RENO. It is able to detect multiple packet losses and thus is much more efficient that RENO in the event of multiple packet losses.

D. TCP Sack

TCP with Selective Acknowledgments (SACK) [11] is an extension of TCP Reno and it works around the problems face by TCP RENO and TCP New-Reno, namely detection of multiple lost packets, and retransmission of more than one lost packet per RTT.

E. TCP Vegas

Vegas is a modified Reno. TCP Vegas [12] is different from TCP Reno in the sense that:

1) A new retransmission mechanism is used
2) An improved congestion avoidance mechanism that controls buffer occupy
3) A modified slow start mechanism

That solves the problem of coarse gain timeout. TCP Vegas include a modified retransmission strategy that is based on fire-gained measurements of the RTT as well as new mechanism for congestion detection during slow start and congestion avoidance.

VI. Research Methodology

In this section, we provide a description for the research methods used for analyzing the performance of TCP variants when the network experience reordering. Network simulation [13] is a technique where a program models the behavior of a network. Network simulator is a software program that simulates the working of a computer network. In simulators, the computer network is typically modeled with devices and traffic before the performance is analyzed. Simulations provide important methods to compare different TCP variants and reordering tolerant algorithms when the network experience packet reordering and study their performance in terms of throughput, congestion window, fast retransmission ratio, link utilization, and timeout ratio. For performance evaluation task, the task can be classified to several steps. Fig. 2 shows software simulation research methodology preferred, that we used, for use a seven-step approach in conducting a successful simulation study adopted by A. M. Law [14].

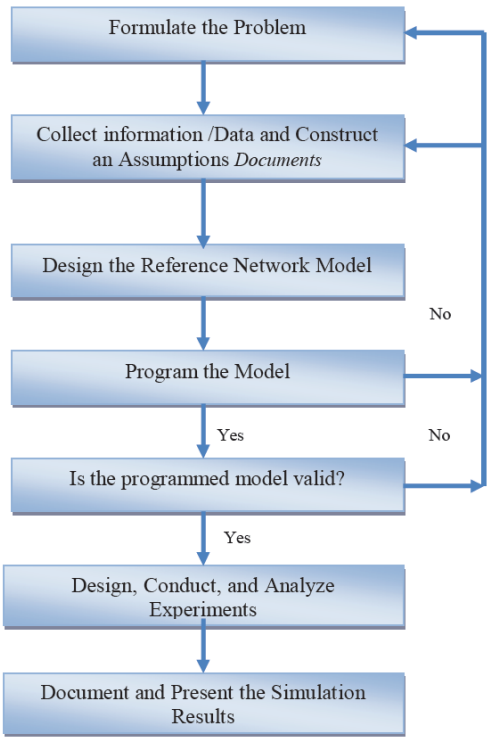

Fig. 2. A seven step approach for conducting a successful simulation study (Law, 2008).

Now, we study the impact of reordering on the various TCP protocols namely TCP Reno, TCP New Reno, TCP SACK, and TCP Vegas. Firstly we analyze the throughput performance of Reno, New Reno, Vegas, and SACK when packets get delayed due to the various reordering distributions such as frequent delay distribution, then we analyze the performance of these protocols when their Dupthresh values are increased in order to prevent false fast retransmissions and unnecessary CWND reductions. Our network topology is given in the following Fig. 3.

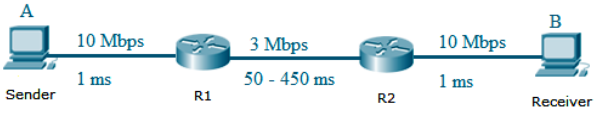

Fig. 3. Network topology.

The simulated network has a source and destination node connected to two intermediate routers. The nodes are connected to the routers. To simulate packet reordering, we repeatedly change the R1-R2 path delay according to normal distribution such that every path delay sample is at least 50ms. The mean and standard deviation of the path delay are (200 α + 50)ms and 200 α /3ms, respectively, where α is the path delay factor ranging from 0 to 2 in our study. A larger α will induce more variation in the path delay, thereby increasing the degree of packet reordering.

The time intervals between two successive changes on the path delay, denoted as inter switching time; regulate the frequency of the reordering events. In our simulation study, the inter switching time, β, is exponentially distributed with mean 50ms or 250ms. The smaller the inters witching time is the more frequently reordering events are produced, and vice versa. Our simulations use 1500 byte packets. We used the drop-tail queuing strategy with a queue size set to 300 segments. Reordering was introduced at the bottleneck link (R1, R2). The experiments were conducted using a single long lived FTP flow traversing the nodes A and B. The maximum window size of the TCP flow was also set to 500 segments. The TCP flow lasts 1100 seconds. We vary the path delay factor from 0 to 2 and set the mean of inter switching time, β to 250 ms to introduce packet reordering events and compare the throughput performance of the simulated network using TCP Reno, SACK, New Reno, and Vegas.

As shown in Fig. 4, Reno and SACK protocols have all very similar throughput performance throughout the tested range of path delay factor, and they give a slight improvement in throughput performance over New Reno for all tested path delay factor, for example, when path delay factor set to 0.2, TCP-SACK throughput performance is almost 2.5% more than TCP New Reno, and when path delay factor set to 0.4, TCP-Reno throughput performance is almost 9% more than TCP New Reno. Because TCP-New Reno uses partial ACK in the Fast Recovery phase, so the packet reordering result in the unnecessary retransmission of an entire window of data, in opposition TCP-SACK and TCP Reno don't use partial-ACK in the Fast Recovery phase, and TCP-SACK uses sack option to detect loss packets. Also TCP SACK and Reno outperform TCP-Vegas for all tested path delay factor from 0 to 2. For example, when path delay factor set to 0.2, TCP-Reno throughput performance is almost 73% more than TCP Vegas, and when path delay factor set to 0.6, TCP Sack throughput performance is almost 31% more than TCP Vegas.

Now, we compare the throughput performance and fast retransmission ratio of the simulated network using Reno, New Reno, SACK, and Vegas when packets experience reordering with different Dupthresh values. The link between the routers is set to 3 Mbps capacity with a configurable delay. The process of delaying a packet is normally distributed with a mean packet delay of 250ms and standard deviation of 200/3ms. Also the inter switching time β is exponentially distributed with mean 250ms. In each experiment we vary the Dupthresh values from 3duplicate ACKs to 18 duplicate ACKs. From the Fig. 5, it is clear when the value of Dupthresh is increased from three duplicate ACKs, the throughput performance of TCP SACK, Reno and New Reno increased to a large extent.

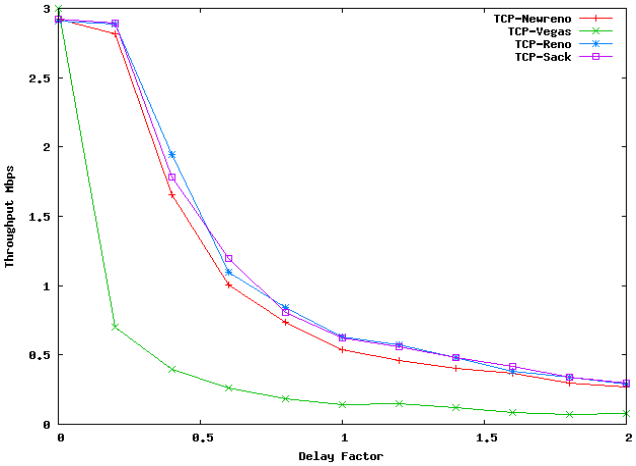

Fig. 4. Frequent delay distributions — throughput versus path delay factor.

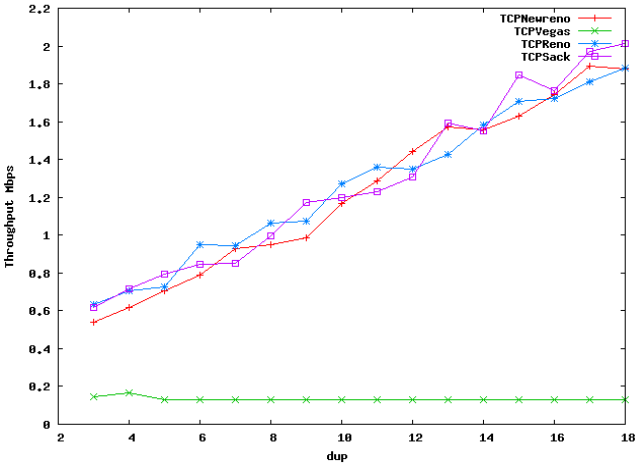

Fig. 5. Frequent delay distributions: throughput versus Dupthrshs.

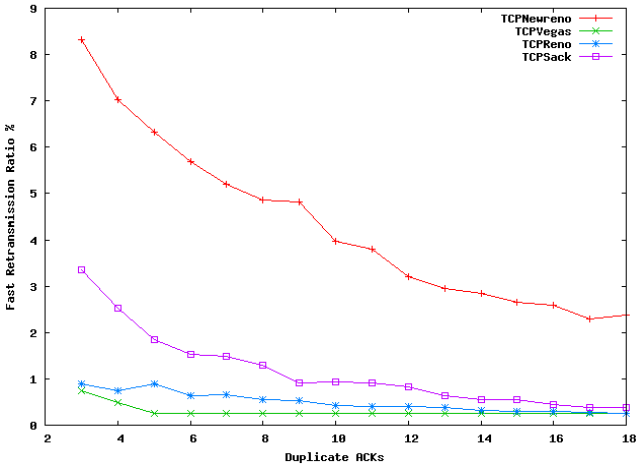

Fig. 6. Frequent delay distributions: fast retransmission ratio versus Dupthrshs.

When the value of Dupthresh equal to 3 the throughput performance of TCP SACK and Reno equal to 0.6 Mbps and the throughput performance of TCP New Reno for this value of Dupthresh equal to 0.54 Mbps, whereas the throughput values of these protocols equal to 1.1Mbps, 1.2Mbps and 1.1 Mbps respectively when Dupthresh equal to 10, and the throughput values of these protocols equal to 2 Mbps, 1.8 Mbps and 1.8 Mbps when Dupthresh equal to 18, Whereas the throughput performance of TCP Vegas doesn't affect with increasing of Dupthresh values.

From the Fig. 6, it is evident that when the value of Dupthresh is increased from three duplicate ACKs, the

number of CWND reductions due to fast retransmissions decrease, and this result in throughput improvement. For example, when the value of Dupthresh equal to 3, the percentage of Fast Retransmitted packet for TCP New Reno equal to 9% and the percentage of Fast Retransmitted packet for TCP SACK for this value of Dupthresh is almost 3.5%, whereas the percentage of Fast Retransmitted packet of these protocols equal to 4% for TCP New Reno and equal to 1% for TCP SACK when the value of Dupthresh equal to 10.

VII. Conclusion

In this research, we evaluated the performance of TCP variants namely TCP SACK, Reno, New Reno, and TCP Vegas when the network experiences reordering and/or reordering. We found that when the network suffer from reordering, the performance of all TCP variants dropped extremely, we also found that when we increase the value of Dupthresh for TCP variants the throughput performance of these protocols can be improved in presence of reordering.

References

[1] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 5th ed. Addison Wesley, 2009.

[2] G. Iannaccone, S. Jaiswal, and C. Diot, "Packet reordering inside the Sprint backbone," Sprint ATL Technical Report TR01-ATL-062917, 2001.

[3] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 IP backbone," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, February 2007.

[4] J. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789-798, Dec. 1999.

[5] J. Postel, "Transmission control protocol," RFC 793, Protocol Specification, DARPA Internet Program, Sept. 1981.

[6] A. Behrouz, *TCP/IP Protocol Suite*, 4th ed. McGraw Hill, 2010.

[7] K. Fall and K. Varadhan, *The NS Manual (Formerly NS Notes and Documentation)*, The VINT Project, May 2006.

[8] A. M. Law, "How to build valid and credible simulation models," in *Proc. the 40th Conference on Winter Simulation*, Florida, 2008.

[9] W. Stevens, "TCP slow start, congestion avoidance fast retransmit algorithm," IETF RFC 2001, January 1997.

[10] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control," in *Proc. ACM SIGCOMM*, 1996, pp. 281-291.

[11] B. Qureshi, M. Othman, and N. A. W. Hami, "Progress in various TCP variants," in *Proc. 2nd International Conference on Computer, Control and Communication*, February 2009, pp. 1-6.

[12] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 8, pp. 1465-1480, 1995.

[13] A. Gurtov and S. Floyd, "Modeling wireless links or transport Protocols," *ACM Computer Communication Review*, vol. 34, pp. 85-96, 2003.

[14] S. Kadry, I. Kamar, A. Kalakech, and, "TCP: An improvement on TCP protocol," *Journal of Theoretical and Applied Information Technology*, pp. 165-171, 2005.

Seifedine Kadry is an associate professor at American University of the Middle East, Faculty of Engineering. He got his master degree in computer science and applied math from AUF-EPFL-Inria, Lebanon in 2002. He received the doctor degree from the Clermont Ferrand II University, France in 2007. His research interests include software testing and security.

Ali Edan is a graduate student from Arts Sciences and Technology University in Lebanon. His research interest is analysis of networking protocol.