A Simulation Analysis of the TCP Control Algorithms

Georgi Kirov

Abstract: *The paper focuses on the different congestion control mechanisms implemented by the Transmission Control Protocol (TCP). The publication presents an experimental estimation of the following TCP control algorithms: Slow-Start and Congestion Avoidance without Fast Retransmit, Tahoe that includes Fast Retransmit and Fast Recovery, and Reno using a modified version of the Fast Recovery. The state of the art of the TCP control approaches is described. The advantages and drawbacks of the above-mentioned algorithms are investigated through the simulation analysis. The TCP performance analysis is based on different scenarios of the network simulation with low percentages of the packet loss.*

Key words: *TCP Control Algorithms, Network Simulation, Network Performance.*

INTRODUCTION

The state of the art of the network congestion shows that it is very difficult problem because there is no way to determine network condition. The congestion occurs when there is a lot of traffic in the networks [1, 2, 5]. The congestion control can be defined as a multicriterial optimization task that has to estimate the following uncertain input parameters: number of users and applications that use the network, network capacity, congestion points, etc.

Rapidly increasing bandwidths [3] and great variety of software applications have created a recognized need for increased attention to TCP flow-control mechanisms [10]. In order to optimize the network utilization the expert researches are focused on the managing of TCP flow-control window. The main purpose of the paper is to analyse the abilities of the TCP control mechanisms for automatically and dynamically defining of optimum window size for a given connection.

The paper is organized as follows: the sections from 1 to 4 describe the basic principles of the different TCP control algorithms, the next section presents a simulation research of the TCP congestion control approaches, the last section presents a comparative analysis of the simulation results and concluding remarks.

TCP CONTROL ALGORITHMS - SLOW START

Over the past several years, TCP is the most used transport protocol all over the world [8, 10]. It is the basic transport protocol for Internet. In this section are described some modifications of TCP that distinguish themselves by their congestion-control algorithms [6].

To resolve the congestion problem the slow start algorithm has been involved. The basic of this approach is the notion of a congestion window (cwnd). When the new connection is established the cwnd is initialized to one packet. Every time a packet with sequence number n arrives at the receiver, the receiver confirms the packet n by sending an acknowledgment (ACK) packet. It contains the information for the sequence number of another packet, which it is waiting for (may not be "n+1") back to the sender. TCP uses an arrival of ACK as a trigger of new packet transmission, i.e. each time an ACK is received, the congestion window is increased by one packet [6, 10]. The sender stops increasing the window size when one reach the limit of the network capacity. The limit is defined as minimum of window that sender can transmit and window that receiver can receive.

Figure 1 shows that when TCP starts a connection between the sender and receiver the window size is set to one packet and the sender waits for its ACK. In the next step, after the ACK receiving, the congestion window is set from one to two, and two packets can be sent. In the case that each of those two packets is received correctly, the congestion window is increased to four. This can be estimated as an exponential growth.

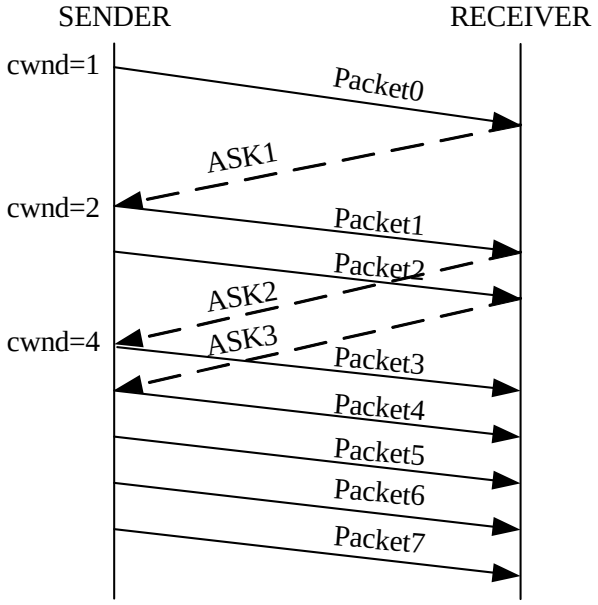

Figure 1. Congestion window increasing

CONGESTION AVOIDANCE

Congestion avoidance is the algorithm that tries to solve the problem with lost packets. The congestion occurs when the rate at which packets arrive at routers is more than routers can send [5]. In general, there are two indications of packet loss: a timeout occurring and the receipt of duplicate ACKs.

Congestion avoidance [7] and slow start are different control algorithms that work together. The combined control mechanism introduces two parameters to adjust the amount of data being injected into the network: a congestion window (cwnd), and a slow start threshold size (ssthresh). The window size is defined by the following formula:

$$Window\ size = \min(advertized\ window, cwnd) \quad (1)$$

where cwnd is a window that sender can transmit, advertised_window is flow control window, which is sent from receiver side.

When the new connection starts, TCP sets cwnd to one packet, ssthresh to arbitrary high value (65535 bytes), and starts slow start mode. In the case of a congestion (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size is saved in ssthresh. and cwnd is set to one packet (i.e., slow start). TCP triggers Slow start at the beginning of a transfer and the window exponentially increases: send one segment, then two, then four, and so on every time an ACK is received. The TCP works in the slow start mode until window size reaches ssthresh [10]. After that TCP performs congestion avoidance. It dictates that cwnd will be incremented by segsize*segsize/cwnd each time an ACK is received, where segsize is the segment size and cwnd is maintained in bytes. This is a linear growth of cwnd, compared to slow start exponential growth (Figure 2) [9].

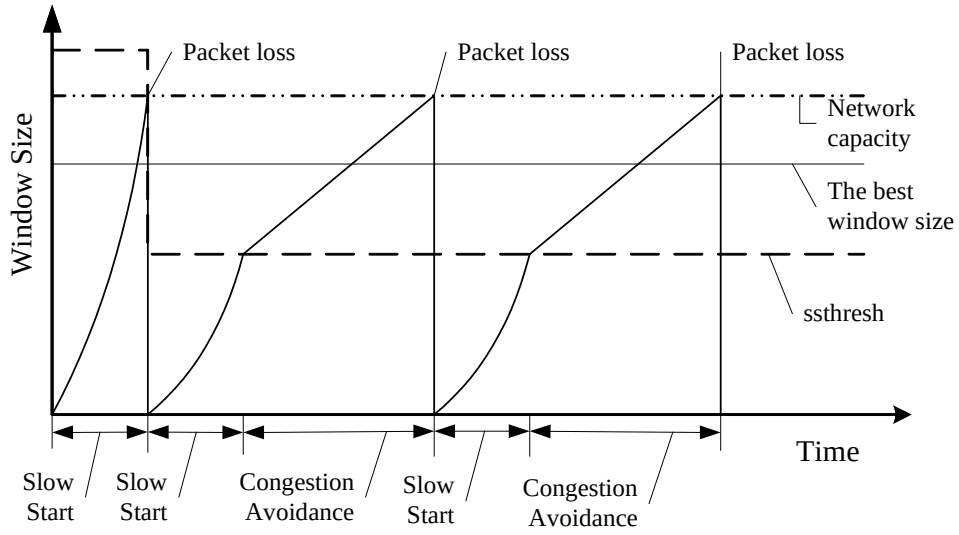

Figure 2. Slow start and Congestion avoidance

FAST RETRANSMIT

The old TCP detects the network congestion and lost packets by the timeout mechanism. When a packet is sent, TCP sets up its own timer to the retransmission timeout period (RTO) for this packet. If receiver correctly receives packet, TCP generates an immediate acknowledgment (ACK) corresponding to the data packet before the timer is expired. TCP assumes that the network is OK. After that TCP automatically informs the timer of the received ACK packet and waits for the other ACK packets. In the case, that TCP doesn't receive required ACK within RTO period, the sender will retransmit the packet whose timer is expired. Further, TCP starts slow-start and sets cwnd to 1 and ssthresh to (old cwnd / 2) (Figure 3).

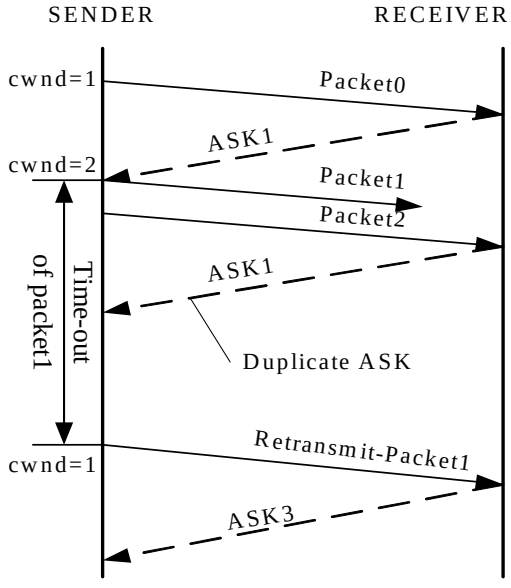

Figure 3. Lost packets by the RTO mechanism

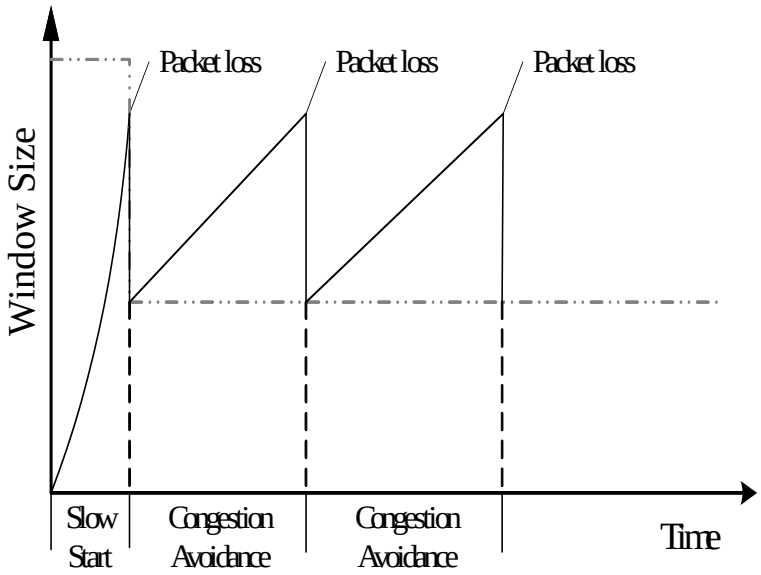

Figure 4. Reno fast recovery

Fast retransmit algorithm retransmits packet without waiting for retransmission timeout. It uses the ability of TCP to return the ACK if the packet is correctly transmit. If the received packet n is out of order the receiver acknowledges the packet n+1 by duplicate again ACK for the wrong packet n. The purpose of the duplicate ACK is to inform the sender that a segment was received is out of order. Therefore, fast retransmit uses duplicate to trigger retransmission packets.

The duplicate ACK can be generated by packet loss or packet reordering. In the case of a reordering only one or two duplicate ACK will be generated before the reordered packet is received. Then the next ACK will be return with the sequence number of another waited packet. The TCP triggers the fast retransmit algorithm when TCP generates three or more duplicate ACK.

FAST RECOVERY

Fast recovery regards the stage after the moment of the congestion. In the last several years some modifications of the TCP fast recovery algorithm have been involved (Tahoe, Reno, Vegas) that distinguish themselves on the basis of their congestion-control mechanisms. The Thaoe's algorithm [4, 10] operates as follows (Figure 2):

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

The other variant of the fast recovery is supposed by Reno. In comparison with the above algorithm it has following differences [4]:

- If the packet loss is caused by RTO (congestion is serious) window size is set to 1 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, but not slow start is performed (Figure 4), i.e. window size is set to old_window_size/2.

SIMULATION OF THE TCP CONTROL ALGORITHMS

In order to investigate the TCP performance and verify the above considerations, the performance of the algorithms is compared through the analysis of the simulation results. For the purpose of simulation is used OPNET IT Guru simulator (Figure 5). The simulation network consists of two subnets (subnet_sender and subnet_receiver) and IP Internet cloud that are connected with PPP_DS3 connections. The elements of the sender subnet are server, router, and 100_BaseT link. The server supports the FTP service. The subnet from receiver side consists of router, 100_BaseT link and ftp client that uses the server supported service. The user can set a lot of TCP parameters. In the simulation the congestion window size and sent segment sequence number are investigated. Depending on initial network conditions different scenarios for the TCP control algorithms are simulated:

- Slow-Start and Congestion Avoidance without Fast Retransmit;
- Tahoe: includes Fast Retransmit and Fast Recovery;
- Reno: adds modification to Fast Recovery.

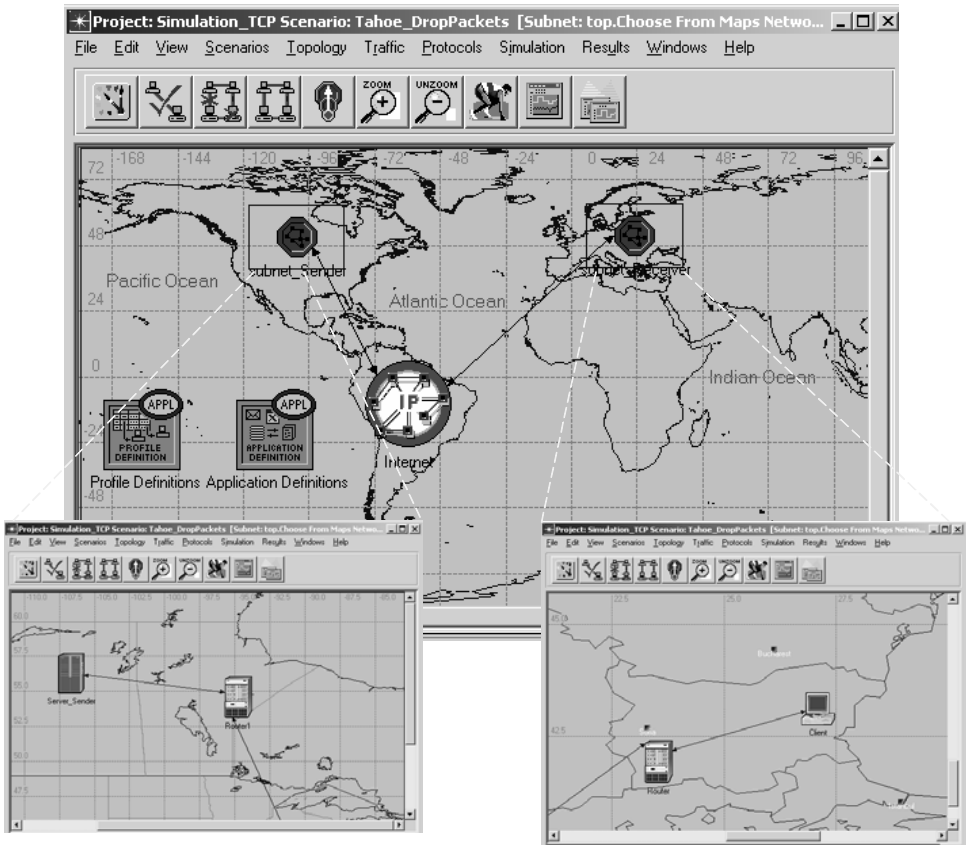

Figure 5. Simulation model

In this scenario the congestion window size and sent segment sequence number of the TCP control algorithms are studied when the packet loss is low (1%). The Figure 6 illustrates simultaneously the performance and the congestion window size of the Tahoe and Reno mechanisms. It confirms the consideration in the chapter 2 that Reno is more effective.

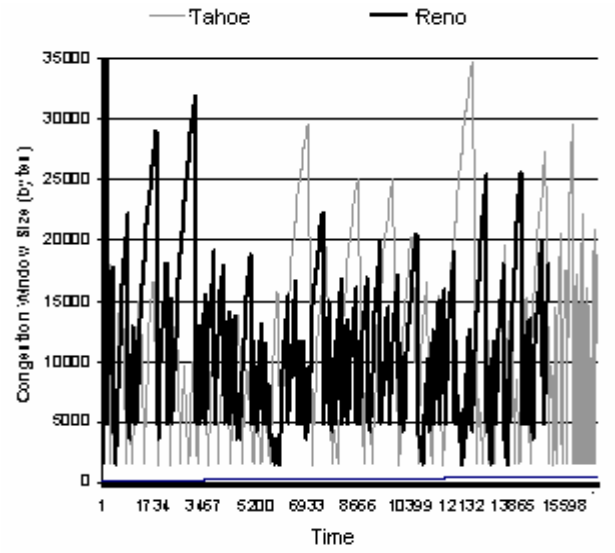

Figure 6. Congestion window size comparison (Tahoe and Reno)

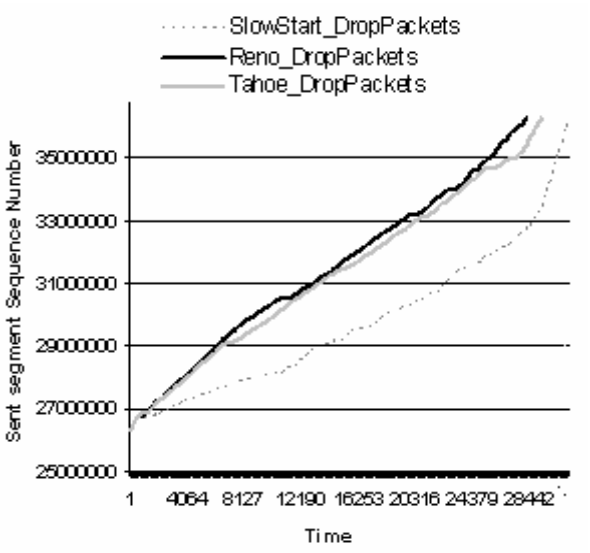

Figure 7. Segment sequence number

The Figure 7 shows the sent segment sequence number in respect to the time. It depicts that Reno is slightly better than Tahoe algorithm. The worst is Slow Start without Fast Retransmit support.

ANALYSIS OF THE SIMULATION RESULTS AND CONCLUSIONS

The paper presents a simulation analysis of the following TCP control mechanisms: Slow-Start and Congestion Avoidance without Fast Retransmit, Tahoe, and Reno. The performance of the algorithms through the analysis of the simulation results is estimated.

The bad result for the Slow Start and Congestion Avoidance without Fast Retransmit can be explained with the fact that the algorithm does not count the duplicate ACKs in order to determine if a packet has been lost. The sender infers that a packet has been lost only when the retransmission timer expires.

The Tahoe algorithm shows better performance. It implements Slow Start, Congestion Avoidance and Fast Retransmit. The last one is triggered when sender receives three duplicate ACKs. The Tahoe method is faster because it retransmits packet without waiting for retransmission timeout. It leads to an optimization of the channel utilization.

The results for Reno are slightly better than Tahoe. The advantage of the Reno algorithms in comparison with Tahoe one is when packet loss is detected, the window size is reduced to one half of the current window size and the congestion avoidance, but not slow start is performed.

REFERENCES

[1] Douglas, C., Internetworking with TCP/IP (2), Prentice-Hall, Inc., 1991.

[2] Douglas, C., Internetworking with TCP/IP (3), Prentice-Hall, Inc., 1991.

[3] Ewerlid, A., Reliable communication over wireless links, in Nordic Radio Symp. (NRS), Sweden, Apr. 2001.

[4] Fall, K., S. Floyd, Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. Computer Communication Review, 26(3), July 1996, pp. 5-21.

[5] Firoiu, V., M. Borden, "A study of active queue management for congestion control," in Proc. IEEE INFOCOM, March 2000.

[6] Jacobson, V., Congestion Avoidance and Control, Computer Communication Review, 18(4), August 1988, pp. 314-329.

[7] Jacobson, V., Congestion Avoidance and Control, in Proceedings of SIGCOMM '88 Workshop, ACM SIGCOMM, ACM Press, Stanford, CA, 1988, pp. 314-329.

[8] Padhye, J., S. Floyd, On Inferring TCP Behavior", Computer Communications Review ACM-SIGCOMM, Vol. 31, August 2001.

[9] Schilke, A., TCP over Satellite Links, Seminar Broadband Networking Technology, TU Berlin, 1997.

[10] Stevens, W., TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, 1999, http://www.faqs.org/rfcs/rfc2001.html

ABOUT THE AUTHOR

Research Fellow, Georgi Kirov, PhD, Institute of Control and System Researches, Bulgarian Academy of Sciences, Phone: +359 2 8700337, E-mail: kirov@icsr.bas.bg.