World Applied Sciences Journal 7 (Special Issue of Computer & IT): 75-79, 2009
ISSN 1818-4952

A Method for Calculating the TCP Utilization for TCP Reno and Tahoe using Markovian Model in WLAN

Mohammad Mehdi Hassani and Sanaz Pashmforush

Islamic Azad University Ayatollah Amoli Banch, Iran

Abstract: This paper presents two analytical models for TCP Reno and TCP Tahoe. For each model an algorithm is derived to calculate the utilizations and packet drop rates. The accuracy of the models is verified by comparing the calculated results versus simulation results. These results show that the TCP Reno is superior to Tahoe by having higher percentage of utilization and lower percentage of packet dropping rate.

Key words: TCP . TCP/IP . congestion control . Tahoe . Reno

INTRODUCTION

TCP is known to be the most useful transport layer protocol all around the globe [1, 2]. It provides a safe and connection oriented service on the network. Varieties of applications on the internet depend on the TCP efficiency and therefore, the analysis of the TCP efficiency is important for conjecture of this usage. Network congestion is one of the factors that reduces the TCP efficiency particularly, when variety of traffics are on the network. Certain input parameters, such as: the number of users, different network applications, network capacity and etc. are used to control the congestion. As the network bandwidth increases [3] and various network applications are created, more attention is paid on the congestion control mechanisms by focusing on the flow control management on the TCP. The main objective of this paper is to analyze the existing congestion control algorithms by Markov model and compare their utilization and throughput. Initially, the existing TCP congestion control algorithms are explained. After this, two generation of TCP protocols; Tahoe and Reno [4] which employ these algorithms are explained. Finally, these two protocols are explained with Markov model and an algorithm is obtained for each protocol to calculate their utilization. The calculation results are verified by simulation and these two TCP protocols are assessed based on these results.

CONGESTION CONTROL ALGORITHMS

Tahoe utilizes Slow-Start and congestion avoidance algorithms while Reno employs these two as well as fast recovery and fast retransmit. Therefore, it is necessary to explain these congestion control algorithms and their functionality.

Slow-start: In this algorithm for every new connection a congestion window (CWND) is initialized in the sender side with a default value. If the receiver, receives a packet with a sequence number (n), it sends an ACK packet (acknowledge) to verify its reception as well as the sequence number of the next packet (n+1) that sender would send. The sender increments the CWND after each ACK packet [1, 5]. In fact, the CWND represents the number of packets that a sender can sequentially send without waiting for their verification. This mechanism tends to an exponential increase in the CWND in the course of time as graphically shown in Fig. 1. However, there is a limit for the increase because the CWND is reduced after each unsuccessful transmission (a packet loss). A

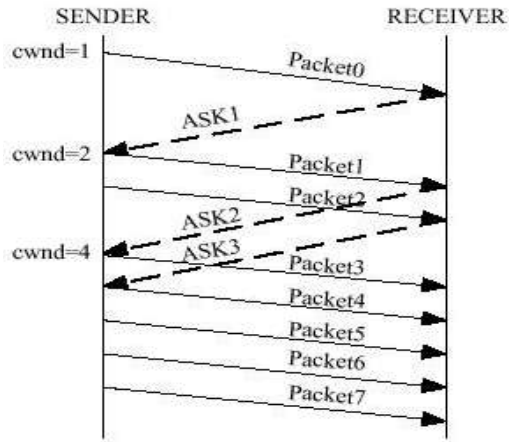

Fig. 1: Increase size of CWND [4]

Corresponding Author: Dr. Mohammad Mehdi Hassani, Islamic Azad University Ayatollah Amoli Banch, Iran

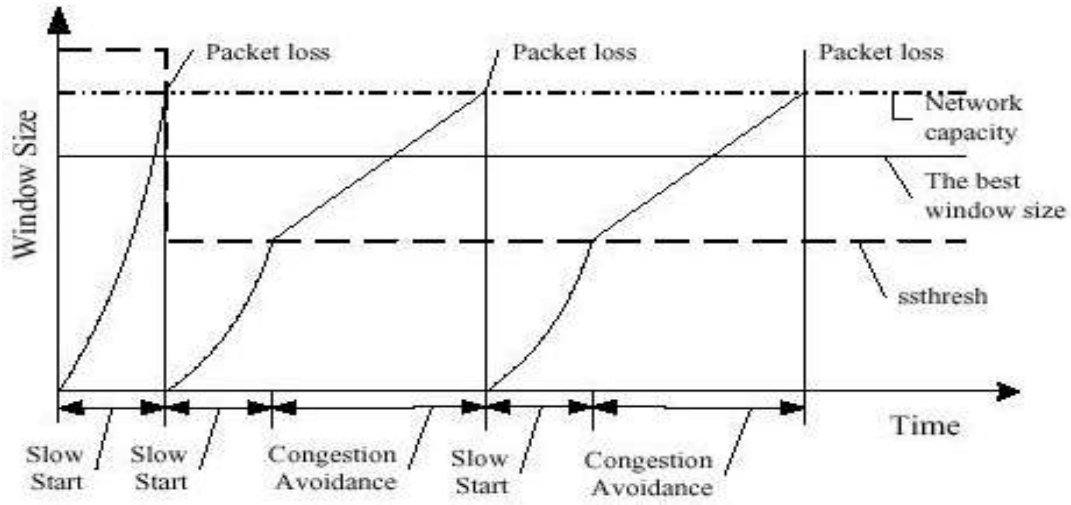

Fig. 2: Fast retransmit algorithm

packet loss is assumed in the TCP by either a time out or duplicate ACK response for the same packet (a dupack).

Congestion avoidance: This algorithm is used for solving the problem of the lost packets when the rate of incoming packets to a router exceeds from its routing speed [4].

Although Congestion Avoidance [6] and Slow-Start are different control algorithm [3] but their performance depends on each other. The control mechanism that made by mixing these two algorithms use CWND and Slow-Start threshold Size (SSTHRESH) to specify the number of sent packets.
The number of sent packets can be obtained from Equation (1)

$$\text{Min (Receive Window, CWND)} \quad (1)$$

where, the receive window is the size of the buffer in the receiver. After connection, the TCP sets CWND to one and the slow-start algorithm will start. When congestion occurs, half of the current window is stored in the SSTHRESH and CWND will be set to one again. TCP continues slow-start algorithm as long as the size of CWND is smaller than SSTHRESH. After CWND exceeds than SSTHRESH the TCP starts the congestion avoidance algorithm, in which CWND increases linearly according to the Equation (2).

$$CWND^{new} = \frac{segmentsize^2}{CWND^{old}} \quad (2)$$

In Equation (2) the segment size equals to the number of packets. Figure 2 shows the operation of

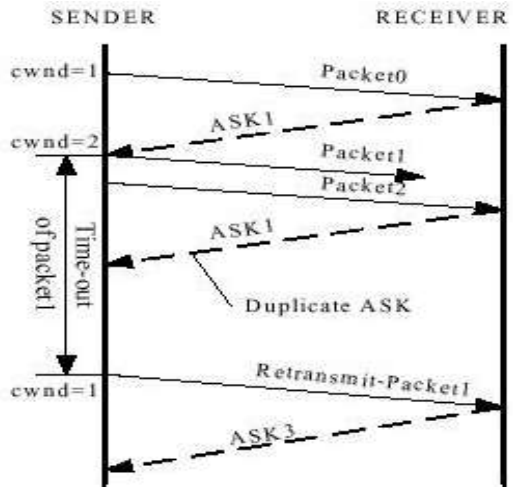

Fig. 3: Fast retransmit algorithm in packet loss

combined slow-start and congestion avoidance algorithms.

Fast retransmit algorithm: Old TCPs would recognize the lost packets and the network congestion by a timeout mechanism. After sending a packet, the receiver waits for a period of time (RTO). During this period, if the receiver receives the packet correctly, sends an ACK response back but if a packet is received out of sequence, the receiver sends two consecutive ACKs (a dupack). A dupack can also be received as a result of a lost packet. In the former more than one dupack (based on the sequence number) might be sent. Receiving more than two dupack indicates a packet loss, as we can see in Fig. 3. Upon receiving a dupack, the TCP retransmits the lost packet without waiting for the end of RTO.

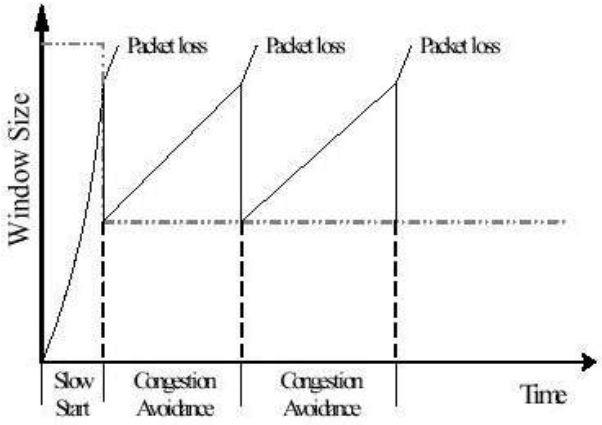

Fig. 4: Fast recovery algorithm

Fast recovery algorithm: In this algorithm, when the sender receives more than two dupacks, assumes a packet loss, calculates the value of SSTHRESH and CWND sequentially from Equations (3) and (4) and sends the packet again.

$$SSTHRESH = Min\ (CWND^{old}, \text{Receiver's advertised window})/2(\text{at least 2 MSS}) \quad (3)$$

$$CWND^{new} = SSTHRESH + \text{number of dupacks} \quad (4)$$

Upon receiving the ACK for this packet, the sender returns the SSTHRESH to the original CWND value. This algorithm acts similar to the slow-start algorithm on facing the timeout problem [7]. Figure 4 shows the window size growth when a combination fast recovery and the fast retransmit is used.

TCP Tahoe: Tahoe is one of the of TCP variations that is suggested by Jackson [8]. This algorithm starts with slow-start before CWND exceeds SSTHRESH. This congestion avoidance causes the CWND to increase linearly. In case of a timeout, the CWND is initialized to one, the SSTHRESH with CWND/2 and the algorithm backs to the Slow-Start. In receiving three dupacks the fast retransmit algorithm is invoked, sends the packet before the end of RTO and immediately returns to the condition that happens after a timeout.

TCP Reno: This TCP variation is essentially same as the Tahoe except using the fast recovery algorithm [9] when three dupacks is received.

Analytical models: In this section the efficiency of the mentioned TCP variations are measured with their Markov models. The model represents the upstream of a mobile which uploads data to a server through an access point and receives ACKs in the presence of downstream packets from the same server and the same node. Therefore the receiving ACKs and the downstream packets use the same buffers. Each state of the Markov model shows the size of the transmit window (CWND) and it is increased exponentially. Therefore, Pi represents the state in which TCP window size is 2^i.

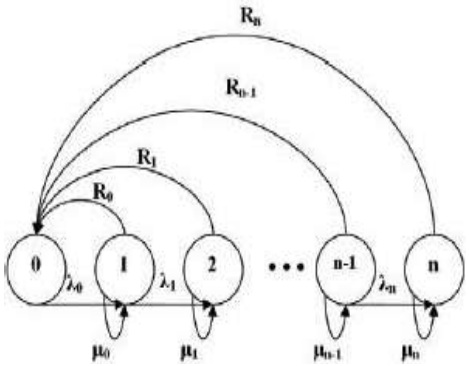

Fig. 5: Markov chain for TCP Tahoe

Analyzing TCP Tahoe: Figure 5 shows our derived Markov model for the TCP Tahoe. In the state "*i*" either all packets are successfully sent or a timeout happen. In the former, the windows size (CWND) is incremented to 2^i and the system goes to the next state while in the latter the size of window returns to one (entering $state_0$). In here, we assume $\lambda = 2^i$ is the packet arrival rate, the U_i is the service rate, the R_i is a rate that is defined by the inverse of the time that receiver is waiting for the ACK (RTO), which can be obtained from (5).

$$R_i = \frac{1}{2^i RTT} \quad (5)$$

Now by using of marcovian model we create a formal with n+1 passive that it gain from (6)

$$flow_{in} = flow_{out} \quad (6)$$

$$(r_{0+}\mu_0)p_1 + (r_1 + \mu_1)p_2 + (r_2 + \mu_2)p_3 + ...$$
$$+ (r_{n-1} + \mu_{n-1})p_n + (r_n + \mu_n)p_{n+1} = (\lambda_0)p_0$$
$$\lambda_0 p_0 = (\lambda_1 + \mu_0 + r_0)p_1$$
$$\lambda_1 p_1 = (\lambda_2 + \mu_1 + r_1)p_2$$
$$\lambda_2 p_2 = (\lambda_3 + \mu_2 + r_2)p_3$$
$$\lambda_3 p_3 = (\lambda_4 + \mu_3 + r_3)p_4$$
$$.$$
$$.$$
$$\lambda_n p_n = r_n p_{n+1}$$
$$p_0 + p_1 + p_2 + p_3 + ...+ p_n + p_{n+1} = 1$$

where RTT is the round a trip time, which for the simplest case is assumed to be one time unit. An iterative algorithm is derived based on this model to calculate the efficiency of this TCP variation. The algorithm shown in (7) and (8) tries to solve a system of n+1 equations with n variables by assuming $k = 2^n$:

$$\begin{aligned} &Pn = \frac{1}{2^{2n}} * P_{n+1} \\ &\text{for}(k = n \text{ to} 0)\{ \\ &l = \frac{1}{k-1} + \sqrt{\frac{2\alpha}{3*wz^2}} + 2^k \\ &p_{k-1} = \frac{1}{2^{k-1}} p_k \} \end{aligned} \quad (7)$$

The solution in (7) starts with an unknown P_{n+1} and after each iteration all P_k are found as a linear multiple of this unknown. Equation (8) can give P_{n+1} and the rest of the unknowns (P_k) can be obtained respectively.

$$p_{n+1} = \frac{1}{1 + \sum_{k=0}^{k=n} p_k} \quad (8)$$

Analyzing TCP Reno: The derived Markov chain for the TCP Reno is shown in Fig. 6. There are three different possibilities:

- All packets are verified and the value of windows size becomes 2^i and the next state is chosen.
- Three dupacks is received, CWND is halved (equal to 2^{i-1}) the system goes back to the previous state.
- A timeout occur, CWND is set to 1 which returns the system to state$_0$ and Slow-Start algorithm starts.

$$\begin{aligned} &FLOW_{IN} = FLOW_{OUT} \\ &R_0P_0 + R_NP_{N+1} = \lambda_0P_0 \\ &R_1P_2 + \lambda_0P_{0+} = (\mu_0 + R_0 + \lambda_1)P_1 \\ &R_2P_3 + \lambda_1P_1 = (\mu_1 + R_1 + \lambda_2)P_2 \\ &. \\ &. \\ &. \\ &\lambda_NP_N = (\mu_N + R_N)P_{N+1} \end{aligned} \quad (9)$$

Now by using of marcovian model we create a formal with n+1 passive that it gain from (9)

An algorithm can be derived based on this model to calculate the efficiency of the Reno. Similarly, an iterative algorithm is derived based on this model to calculate the efficiency of this TCP variation. The algorithm shown in (9) uses (8) to solve a system of n+1 equations with n variables by assuming $k = 2^n$.

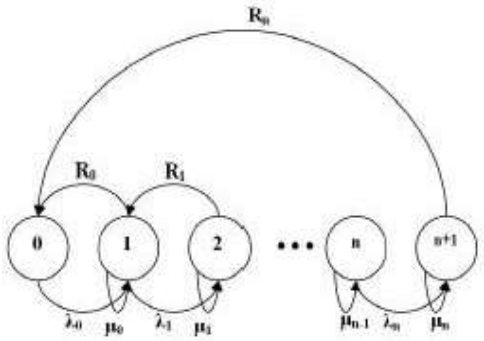

Fig. 6: Marcovian chain TCP Reno

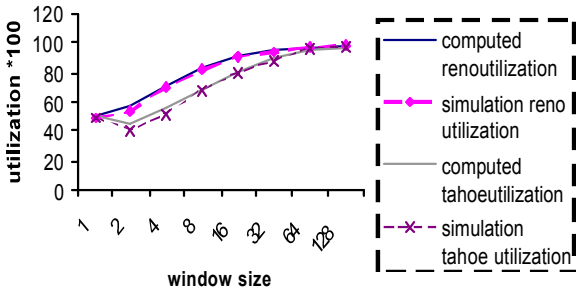

Fig. 7: TCP tahoe and reno utilization

$$\begin{aligned} &p_n = \frac{1}{2^{2n}} p_{n+1} \\ &p_{n-1} = \frac{\frac{1}{2^{n-1}} + 2^n}{2^{n-1}} p_n \\ &\text{for}(i = n - 2 \text{ to} 0) \\ &\quad l = (\frac{1}{2^i} + 2^{i+1})p_{i+1} + (\frac{1}{2^{i+1}})p_{i+2} \\ &\quad p_i = \frac{1}{2^i}\} \end{aligned} \quad (10)$$

The utilization and drop rate that are calculated with these algorithms are shown in Fig. 7 and 8.

Simulation study: In order to verify our analytical model, the utilization of the TCP Tahoe and Reno are obtained by computer simulation. A scenario is simulated using ns2 with a network of two subnets (receiver and sender) that communicate through a server with a base station. The base station buffer size is assumed to be 100 packets. The simulation runs 5 times, each lasting 100 seconds. Figure 7, shows both simulated and calculated utilizations from these two algorithms versus window size. As can be seen, the simulation results closely follow the calculation results as expected. The drop rate versus window size is also found for these two algorithms and the results are shown in Fig. 8. As we see, in the same window size

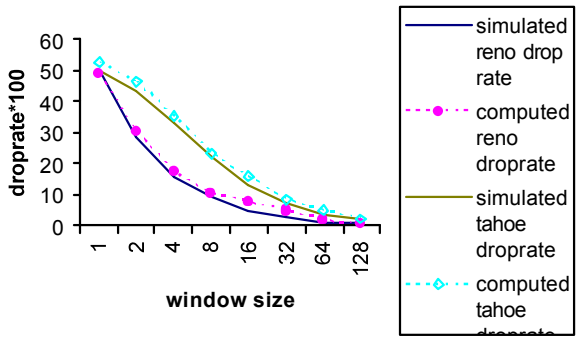

Fig. 8: TCP tahoe and reno drop rate

Reno has better utilization and lower drop rate than of the Tahoe because of using fast recovery algorithm. This means that Reno is more efficient than Tahoe for the same window size. This due to the fact that Reno in facing timeout or packet loss only halves its window size while for the same problems, Tahoe initializes its window to one which takes a longer time to recover.

CONCLUSION

In this paper TCP Reno and Tahoe are explained based on their Markov models. These models are used to derive two iterative algorithms to calculate utilization and drop rate. The calculation results are verified by simulation results. It is shown that TCP Reno outperforms TCP Tahoe because of using fast recovery algorithm.

REFERENCES

1. Stevens, W., 1999. TCP Slow Start, Congestion Avoidance, Fast Retransmit,Recovery Algorithms. RFC 2001, http://www.faqs.org/rfcs/rfc2001htl
2. Padhye, J. and S. Floyd, 2001. On Inferring TCP Behavior, Computer Communications Review ACM-SIGCOMM, Vol: 31.
3. Ewerlid, A., 2001. Reliable communication over wireless links, in NordicNRS, Sweden.
4. Kirov, M.G., 2005. A Simulation analysis of the TCP control algorithm. International Conference on Computer System and Technologies, CompSysTech.
5. Jacobson, V., 1988. Congestion Avoidance and Control. Computer Communication Review, 18 (4): 314-329.
6. Jacobson, V., 1988. Congestion Avoidance and Control. Proceedings of SIGCOMM '88 Workshop, ACM SIGCOMM, ACM Press, Stanford, CA, pp: 314-329.
7. Pilosof, S., R. Ramjee, D. Raz, Y. Shavitt and P. Sinha, 2003. Understanding TCP Fairness over Wireless LAN. IEEE INFOCOM, 2: 863-872.
8. Jacobson, V., 1988. Congestion avoidance and Control. Proceedings of 88 workshop, ACM SIGCOMM, ACM. Press, Stanford, CA, pp: 314-329.
9. Fall, K. and S. Floyd, 1996. Simulation-Based Comparisons of Tahoe, Reno, TCP. Computer Communication Review, 26 (3): 5-21.