# Effect of CWND and RTO in the Performance of Wireless Networks

***Daivik Bhatt***
Dharnsinh Desai University , Nadiad.
*daivikbhatt.10@gmail.com*

***Abhi Shah, Dipen Khatri***
Dharnsinh Desai University , Nadiad.
*Shah.abhi22@yahoo.com,dipenkhatri15@gmail.com*

## ABSTRACT

*In this paper we describe a* ***ns*** *based simulation analysis of TCP performance over a hybrid network with first link as a wireless link. The latter choice is for understanding performance issues with better clarity. The effect of wireless errors on different TCP parameters resulted into end to end performance degradation of TCP is investigated. Based on the analysis of the basic MAC loss recovery mechanism, we are able to show impact of delay introduced by MAC loss recovery over the performance of TCP. With increased MAC loss recovery duration TCP is unable to conceive maximum of available bandwidth over a network. In addition, it also describe that adaption in CWND will improve the performance of wireless networks. All above simulation results clearly show a) scope for improving MAC loss recovery and b) requirement of tuning between TCP and MAC loss recovery, particularly for retransmission mechanism.*

**KEY WORDS:** ACKs; Retransmission Timeout; Medium Access Control; Round Trip Time; Distributed Coordination Function.

## INTRODUCTION

TCP is a large expended transport layer protocol that provides connection oriented, reliable service to the application layer (RFC 793, 1981). In TCP, reliability is achieved by retransmitting lost packets. Thus, each TCP sender maintains a running average of the estimated round trip delay and the average deviation derived from it. Packets will be retransmitted if the sender receives no acknowledgment within a certain timeout interval or receives duplicate acknowledgments.

To understand TCP behavior and improve TCP performance over wireless networks, given these wireless specific challenges, considerable research has been carried out and many schemes have been proposed. As the research in this area is still active and many problems are still wide open, this article serves to pinpoint the primary causes for TCP performance degradation over wireless networks, (Oliveira, R. and T. Braun. 2002) and cover the state of the art in the solution spectrum, in hopes that readers can better understand the problems and hence propose better solutions based on the current ones.

We present in Section 2 a brief overview of TCP congestion control mechanisms. Section 3 starts by identifying the challenges imposed on the standard TCP in one-hop wireless networks. The structure of Section 4 is similar to that of Section 3, except that comparison between TCP performance over wired and wireless links is focused here.

## OVERVIEW OF TCP

Before we dive into the details it is necessary to prepare the reader by presenting an overview of not only the basic functionality of TCP but also the state-of-the-art in TCP. The basic functions of TCP as a transport layer protocol include flow control, error recovery and congestion control, while the state-of-the-art techniques include fast retransmission and recovery, selective acknowledgment, etc., mainly focusing on how to promptly and effectively respond to network congestion.

### Basic Functionality of TCP

It is well known that TCP is a connection-oriented transport protocol that is aimed for guaranteeing end-to-end reliable ordered delivery of data packets over wired networks. For this purpose, basic functionalities such as flow control, error control, and congestion control are indispensable. While these functions have a clean-cut definition of their own, in practice they are closely coupled with one another in TCP implementation.

In TCP, a sliding window protocol is used to implement flow control, in which three windows are used, namely, Congestion Window, advertised window, and Transmission Window. Congestion window indicates the maximum number of segments (Without causing confusion, the term segment and packet are used interchangeably henceforth) that the sender can transmit without congesting the network. As shown next in details on congestion control, this number is determined by the sender based on the feedback from the network. advertised window, however, is specified by the receiver in the acknowledgements. Advertised window indicates to the sender the amount of data the receiver is ready to receive in the future. Normally, it equals to the available buffer size at the receiver in order to prevent buffer overflow. Transmission window means the maximum number of segments that the sender can transmit at one time without receiving any ACKs from the receiver. Its lower edge indicates the highest numbered segment acknowledged by the receiver.

Obviously, to avoid network congestion and receiver buffer overflow, the size of transmission window is determined as the minimum of the congestion window and the receiver's advertised window. To notify the sender that data is correctly received, TCP employs a cumulative acknowledgement (ACK) mechanism. In other words, upon the receipt of an ACK, the sender knows that all previously transmitted data segments with a sequence number less than the one indicated in the ACK are correctly received at the receiver. In the case that an out-of-order segment (identified on the basis of sequence numbers) arrives at the receiver, a duplicate ACK is generated and sent back to the sender. It is important to note that in wired networks, an out-of-order delivery usually implies a packet loss. If three duplicate cumulative ACKs are received, the sender will consider the packet is lost. A packet loss is also assumed if the sender does not receive an ACK for the packet within a timeout interval called retransmission timeout (RTO), which is dynamically computed as the estimated round-trip time (RTT) plus four times the mean deviation. By retransmitting the lost packet, TCP achieves reliable data delivery. It turns out that in wired networks, almost all the packet losses are due to network congestion rather than transmission errors. Thus, in addition to retransmission, TCP responds to packet losses by invoking its congestion control mechanism. TCP congestion control is also based on the sliding window mechanism described above and consists of two major phases: slow start and congestion avoidance. In the slow start phase, the initial congestion window size (cwnd) is set to one maximum segment size (MSS) and is incremented by one MSS on each new acknowledgement, resulting into exponential growth of cwnd per RTT. However it may be a little different from exponential if receiver is implemented with delayed acknowledgement.

After cwnd reaches a preset threshold (ssthresh), the congestion avoidance starts and it is increased linearly, i.e., it is increased by one segment for each RTT. Upon a timeout, ssthresh is set to the half of the current transmission window size (but at least two segments) and the congestion window is reduced to 1 MSS. Then slow start mechanism starts again. This procedure is also called the additive increase and multiplicative decrease algorithm (V. Jacobson, 1988). The entire congestion control algorithm is illustrated in Fig. 1. Note that the sender reacts to three duplicate ACKs in a different way, which is described in fast retransmission and fast recovery.

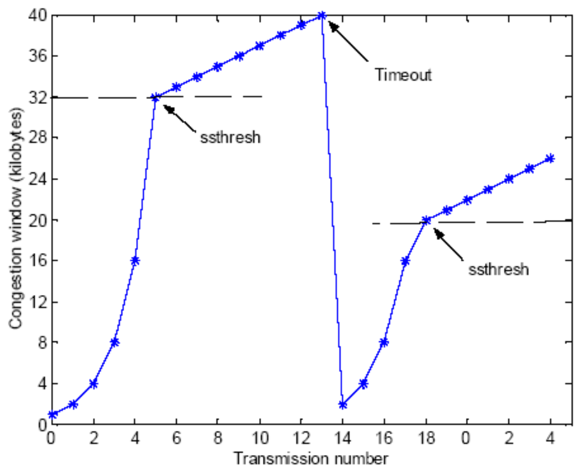

Fig. 1 TCP *cwnd* Dynamics (A. S. Tanenbaum, 2002)

**Fast Retransmission and Fast Recovery**

As noted earlier, a packet can be considered lost if three duplicate ACKs are received. In this case, TCP performs a fast retransmission of the packet. This mechanism allows TCP to avoid a lengthy timeout during which no data is transferred. At the same time, ssthresh is set to one half of the current congestion window, i.e., cwnd, and cwnd is set to ssthresh plus three segments. If the ACK is received approximately one round trip after the missing segment is retransmitted, fast recovery is entered. That is, instead of setting cwnd to one segment and starting with slow start, TCP sets cwnd to ssthresh, and then steps into congestion avoidance phase (W. Stevens, 1997, RFC 2001). However, use of partial acknowledgement in TCP New Reno allows recovery of multiple packet losses in the same window without RTO expiration.

## TCP IN ONE-HOP WIRELESS NETWORKS

In this section, we focus on TCP performance in one-hop wireless networks, which typically include wireless node communicating over a wired LAN through a Base Station. Compared with wired networks, one-hop wireless networks have some inherent adverse characteristics that will significantly deteriorate TCP performance if no action is taken (P. Sinha,1999). In essence, these characteristics include random channel errors, mobility & communication asymmetry. In wireless channels, relatively high bit error rate because of multipath fading and shadowing may corrupt packets in transmission, leading to the losses of TCP data segments or ACKs. If it cannot receive the ACK within the retransmission timeout, the TCP sender immediately reduces its congestion window to one segment, exponentially backs off its RTO and retransmits the lost packets. Intermittent channel errors may thus cause the congestion window size at the sender to remain small, thereby resulting in low TCP throughput. Temporary disconnections, resulting in packet losses and delay are also responsible for throughput degradation due to obvious reasons (R. Caceres, 2001). TCP will suffer a lot if it treats such losses as congestion and invokes unnecessary congestion control mechanisms .

For all of the simulations we used simple network topology implemented under NS as shown below. Link 1 is represented as either a) Wired link or b) wireless link with the specified characteristics. Losses are introduced over link 1 only. For comparison of parameters over wired network with the parameters over a hybrid network standard error model in NS is used for introducing errors on both types. However for understanding behavior over wireless link, we have introduced disconnection losses, caused by providing mobility to MH in and out of the coverage area of BS.

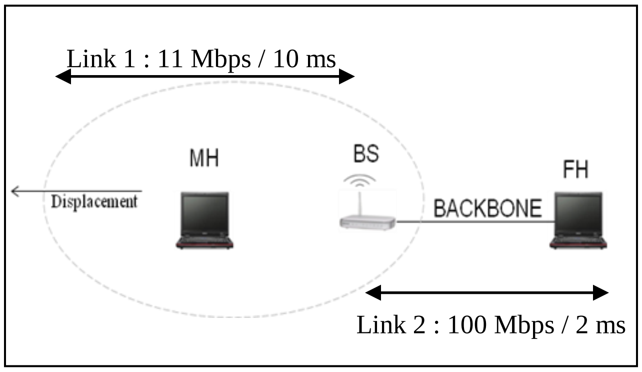

Fig. 2 Network Scenario & Topology

## RETRANSMISSION OVER A WIRELESS LINK

Application data is broken into TCP segments with a 20 bytes header containing among other things the sequence number for ARQ purposes. When expecting an acknowledgement from the other end, a retransmission timer is triggered. The value of this timer is given by the retransmit timeout RTO, which is closely linked to the round-trip time measurement. The round-trip time (RTT) refers to the time it takes to a segment to go from the transmitter to the receiver and to come back. The RTT is estimated based on a smooth algorithm taking into account a few samples of packet time stamps. The recommended value for RTO is roughly 2 RTT. However when TCP transmissions are over an IEEE 802.11 DCF compliant network links, the MAC layer attempts to recover from wireless losses locally, with the help of link layer retransmissions. Compared with TCP retransmissions link layer retransmissions adapt quickly to link characteristics due to shorter timeout periods. Moreover since the length of a frame is much shorter than that of a TCP packet, retransmission in the link layer costs less than that in TCP. This introduces MAC layer retransmission delay at each MAC node, accountable for increased RTT estimation by TCP. To understand the impact of wireless MAC retransmission over an RTT estimation we have analyzed the trace file, generated based on the TCP simulation over a wireless link.

**27.75248** TCP Segment seq_no 10119 transmitted.

After delay 1.52 ms (Processing Delay at node )

**27.75468** 10119 1st Transmission Attempt at MAC
**Failure after MAC Rtx Timeout.**
**27.77688** 10119 2nd Transmission Attempt at MAC
**Failure after MAC Rtx Timeout.**
- - - - - - - - - - - -
**27.83704** 10119 4th Transmission Attempt at MAC
**ACK Received at MAC layer.**

**27.959742** ACK at TCP for seq_no 10119 received.

RTT Estimated is of **240.206 ms ,** where as

The actual RTT is approximately
2 ms of Processing delay at each node for DATA = 6 ms
2 ms of Processing delay at each node for ACK = 6 ms
10 ms + 2 ms of Link delay for DATA frame = 12 ms
10 ms + 2 ms of Link delay for ACK frame = 12 ms
**36 ms**

Fig. 3 Trace file Analysis of MAC Retransmissions

As shown in figure 2 above, each MAC frame will undergo retransmission maximum up to Retry Limit threshold ( say m ). After that the frame is considered to be dropped and hence TCP reaction for retransmission will be triggered. However if the MAC frame transmitted after n transmissions (assume n <=m), TCP will be unaware of retransmissions at MAC layer and therefore estimates RTT based on the timestamps as mentioned earlier. However this RTT estimation will be much higher than the actual link delay, due to additional delay of retransmission attempts, introduced by the MAC node. This will make

TCP estimation of RTT inappropriate. This increase value for RTT inappropriately triggers TCP for reducing its transmission rate, degrading TCP throughput. (Higher RTT is considered as an indication of Network Congestion, Chandran, K, 1997)

## COMPARISION OF RTT ESTIMATION BY TCP

The following figure 3 shows the comparison between RTT estimation by TCP over a network with a) All wired links and b) first link as an IEEE 802.11x DCF compliant link. For both of the above simulations network shown in figure 2, mentioned earlier is simulated. Link between node MH and BS is represented either for case a) or for case b). However the other link characteristics are remained same.

As discussed earlier the local MAC retransmissions over a wireless link increases the RTT estimation and hence resulted into reduced TCP's transmission rate. This will degrade end to end TCP's throughput.

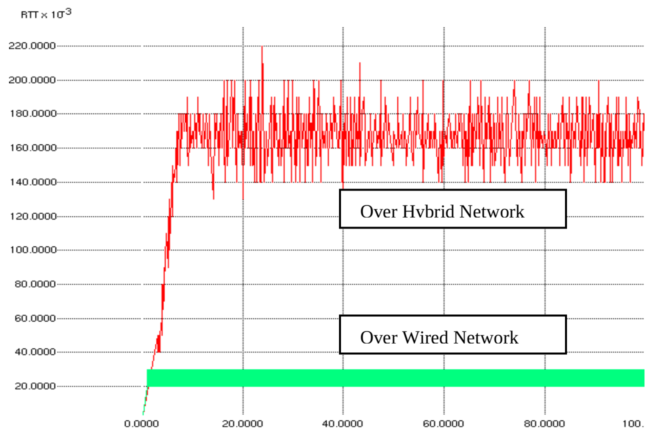

Fig. 4 RTT Comparison over Wired and Hybrid Network

As we know the best strategy to deal with wireless losses is to discover the losses and to transmit quickly as soon as the channel is restored. However increased value of RTT, resulting into higher value for RTO, causes long transmission pauses at TCP particularly after TCP's retransmission timeouts. As shown in figure 5 exponential back off in RTO prevents TCP to re- transmit, immediately after the loss episode.

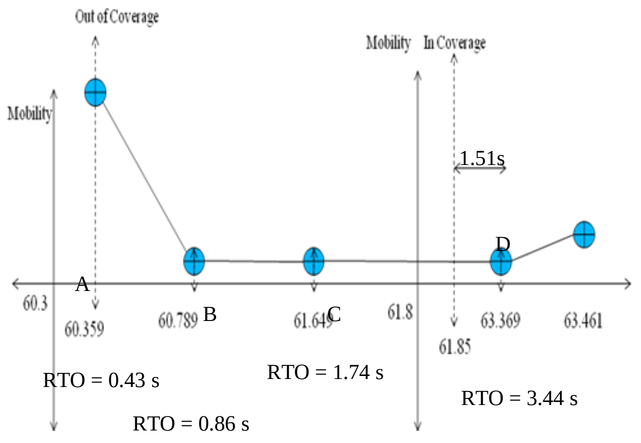

Fig. 5 Effect of Exponential Back off in RTO

The above figure 5 is obtained based on the simulations under NS. The above analysis is for the TCP segment with sequence number 3535. The segment is transmitted at time 59.92945 second having the RTO of apprx. 0.2 sec. Here this higher value of RTO is due to inappropriate estimation of RTT as described earlier. Frequent timeouts causes exponential back off in RTO, introducing long transmission pauses at TCP, badly affecting the TCP's throughput. As seen from the observation, TCP unnecessarily stops its transmission for 1.51s of duration, which is considered to be very high for wireless network applications. In short we may conclude that inappropriate RTT estimation lead to long transmission pauses at TCP.

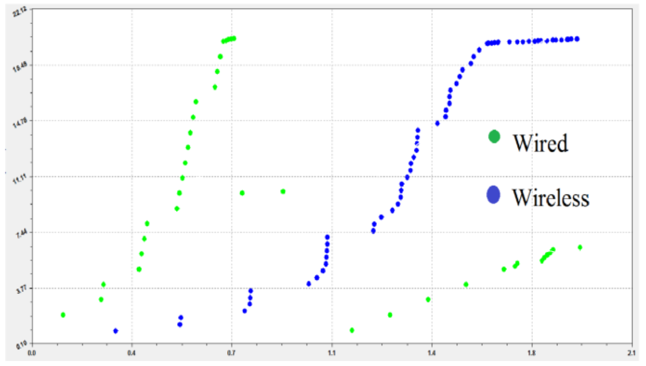

Fig. 6 CWND Comparison between Wired and Wireless Network

The above figure 6 is obtained based on the simulations under NS for wired and wireless network. From above graph shows that the slope for wireless topology is low compared to wired, due to which CWND of wireless network remain in slow start phase because of wrong estimation of RTO. This is one of the reason for degraded performance of wireless topology.

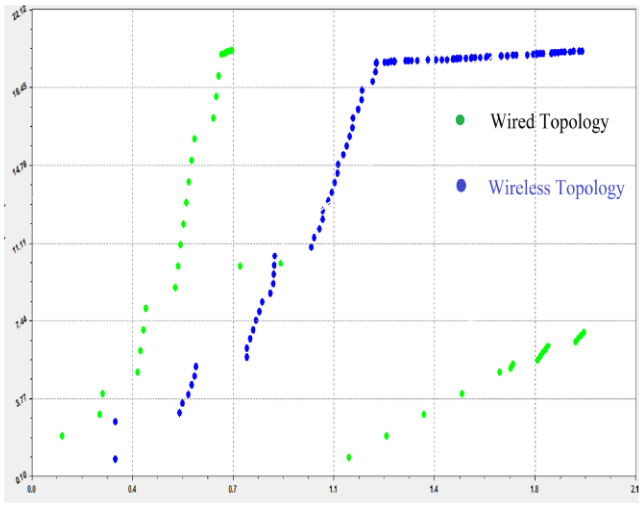

Fig. 7 CWND Comparison between Wired and Wireless Network with higher initial value of CWND for wireless
Figure 7 shows the simulation result of same wired and wireless topology but with increased initial value of CWND for wireless network. Due to higher initial value of CWND the slope of wireless network is high and which TCP performance for wireless network increases.

## CONCLUSION

The goal of this paper is to analyze the behavior of the TCP protocol particularly over a wireless networks characterized by high error rates and to understand why its implementation over a wireless link is not efficient. TCP actually needs to estimate RTT appropriately, may be based on some adaptations in order to be efficient on a wireless environment. This is the main factor causing very large difference in TCP performance over the usual wired network for which TCP was originally designed and the network with wireless component.

Based on the simulations, we derived the followings.

1) RTT estimation by TCP in presence of wireless network segment is very poor. As discussed earlier the rate at which TCP adopts to the capacity of the network ( during slow start phase ), is very less compared to that over a wired network.
2) Increased value of estimated RTT , resulting into higher RTO will cause long wait state for TCP whenever it encounters timeout. This becomes very significant in a network scenario having frequent timeouts, mainly due to loss of retransmissions. The later effect causes long transmission pauses at TCP even after reestablishment of link or end of loss episode.

## REFERENCES

Chandran, K.; S. Raghunathan; S. Venkatesan; R. Prakash. 1997. "*A Feedback Based Scheme For Improving TCP Performance In Ad-Hoc Wireless Networks.*" In *Proceedings of International Conference on Distributed Computing Systems-ICDCS '98* (Amsterdam, The Netherlands, May 26-29). IEEE, 472-479.

Liu, J.; S. Singh. 2001. "*ATCP: TCP for Mobile AdHoc Networks.*" *IEEE Journal on selected areas in communications*, vol. 19, No. 7, July: 1300-1315.

Tanenbaum A. S., 2002, "*Computer Networks*," 4th Edition, Prentice-Hall International, Inc.

Oliveira, R. and T. Braun. 2002. "*TCP in Wireless Mobile Ad Hoc Networks.*" Technical Report IAM- 02-003. Inst. Of Computer Science, University of Berne, July. (available at ftp://ftp.iam.unibe.ch/pub/TechReports/2002/iam-02-03.pdf).
R. Caceres and L. Iftode, July 2001, "*Improving the performance of reliable transport protocols in mobile computing environments*," IEEE JSAC. Vol.19, No.7.

*RFC 793*, 1981, Transmission control protocol, protocol specification,

Sinha P., N. Venkitaraman, R. Sivakumar and V. Bharghavan, Aug. 1999, "*WTCP: a reliable transport protocol for wireless wide-area networks*," MobiCom'99.

V. Jacobson and M. Karels, Aug. 1988 , "*Congestion avoidance and control*," Proceedings of ACM SIGCOMM'88.

W. Stevens, RFC 2001, 1997, "*TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms*".