# Experimental Evaluation of Memory Effects on TCP Traffic in Congested Networks

**Kulvinder Singh[1], Anil Kumar[2]**

**[1]Assistant Professor, Department of Comp. Sc. & Engg. Vaish College of Engineering, Rohtak(Haryana), India**

**[2]Assistant Professor, Department of Comp. Sc. & Engg. Vaish College of Engineering, Rohtak(Haryana), India**

**Abstract**

Today the Internet is a worldwide-interconnected computer network that transmits data by packet switching based on the TCP/IP protocol suite. Internet has TCP as the main protocol of the transport layer. The performance of TCP is studied by many researchers. They are trying to analytically characterizing the throughput of TCP's congestion control mechanism. Internet routers were widely believed to need big memory spaces. Commercial routers today have huge packet memory spaces, often storing millions of packets, under the assumption that big memory spaces lead to good statistical multiplexing and hence efficient use of expensive long-haul links.
In this paper, we summarize the works and present the experimental study result with big memory space size and give a qualitative analysis of the result. Our conclusion is that the, round-trip time (RTT) is not increased by linear, but by quadric when the memory space size of the bottleneck is big enough. Our goal is to estimate the average queue length of the memory space size and develop a TCP model based on RTT and the average queue length.

***Keywords:*** *Traffic management, Congestion control, congestion mechanism, congestion model, memory size.*

## 1. Introduction

The traffic across internet is increasing so congestion is becoming an important issue in network applications. When congestion is occurred the packet round trip time is increased and probability is lost and decreased in the throughput. Transport protocol must deal with this traffic congestion in order to make best use of the capacity of network. TCP adopt a window based congestion control mechanism. Traditionally experimental study and measurement have been the tools of choice for checking the performance of various aspects of TCP. But the amount of non-TCP traffic flows (such as multimedia traffic) keep increasing in today's Internet, non-TCP flows should share the bandwidth with TCP flows "friendly", which means the throughout of the non-TCP flows transport protocol should be approximately the same as the TCP. In order to do that, we must analytically model the TCP behavior and characterize the throughput of TCP's congestion control mechanism.

In this paper, we investigate is the behaviour of a single TCP flow when using memory spaces with different size. We summarize the TCP congestion control mechanism and the method of analytically characterize the throughput of TCP's congestion control mechanism. We present the experimental study result with big memory space size and give a qualitative analysis of the result.

## 2. An overview of Different mechanisms of TCP congestion control

We can distinguish two major kinds of congestion control window-based such as TCP (Transmission Control Protocol), and rate-based that for example regulates ATM-ABR [2] (Available Bit Rate service of an Asynchronous Transfer Mode network) traffic. The window-based congestion control model in the Internet attempts to solve an optimization problem by decoupling the network problem to that of individual source utility maximization by assigning bit-marks.

We have a reliable protocol TCP window-based acknowledgment clocked flow control protocol. The sender control the sending rate by increasing or decreasing the size of its congestion window according to the acknowledgement received. The strategy used for congestion window size adjustment is known as AIMD (Additive Increase Multiplicative Decrease).

### 2.1. Slow Start and Congestion Avoidance

The slow start[8] and congestion avoidance algorithms must be used by a TCP sender to control the amount of outstanding data being injected into the network.
The congestion window (cwnd) is a sender-side limit on the amount of data the sender can transmit into the network before receiving an acknowledgment (ACK), while the receiver's advertised window (rwnd) is a receiver-side limit on the amount of outstanding data. The minimum of cwnd and rwnd governs data transmission. The variables threshold (ssthresh) is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission.
The slow start algorithm is used at the beginning of a transfer, or after repairing loss detected by the retransmission timer.

### 2.2. Congestion Avoidance

In congestion avoidance we deal with lost packets due to traffic congestion. It is described in [4]. The assumption of the algorithm is that packet loss caused by damage is very small; therefore the loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss: a timeout occurring and the receipt of duplicate ACKs. Congestion avoidance and slow start are independent algorithms with different objectives.
But when congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again.

### 2.3. Fast Retransmit

A TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives. The purpose of this ACK is to inform the sender that a segment was received out-of-order and which sequence number is expected. From the sender's perspective, duplicate ACKs can be caused by a number of network problems. First, they can be caused by dropped segments. In this case, all segments after the dropped segment will trigger duplicate ACKs. Second, duplicate ACKs can be caused by the re-ordering of data segments by the network. Finally, duplicate ACKs can be caused by replication of ACK or data segments by the network.
The TCP sender should use the "fast retransmit" algorithm to detect and repair loss, based on incoming duplicate ACKs. The fast retransmit algorithm uses the arrival of 3 duplicate ACKs as an indication that a segment has been lost. After receiving 3 duplicate ACKs, TCP performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire.

### 2.4. Fast Recovery

A TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives. The purpose of this ACK is to inform the sender that a segment was received out-of-order and which sequence number is expected. From the sender's perspective, duplicate ACKs can be caused by a number of network problems. First, they can be caused by dropped segments. In this case, all segments after the dropped segment will trigger duplicate ACKs. Second, duplicate ACKs can be caused by the re-ordering of data segments by the network. Finally, duplicate ACKs can be caused by replication of ACK or data segments by the network. In addition, a TCP receiver should send an immediate ACK when the incoming segment fills in all or part of a gap in the sequence space. This will generate more timely information for a sender recovering from a loss through a retransmission timeout, a fast retransmit, or an experimental loss recovery algorithm.

## 3. Congestion Control and Sizing Router Memory spaces

The goal of this research is to investigate the memory space size that is required in order to provide full link utilization given a used TCP congestion control algorithm. We start by considering the case of a single TCP connection.
We have seen that the standard TCP congestion control algorithm is made of a probing phase that increases the input rate up to fill the memory space and hit network capacity [10]. At that point packets start to be lost and the receiver sends duplicate acknowledgments. After the reception of three duplicate acknowledgments, the sender infers network congestion and reduces the congestion window by half. TCP does not modify TCP behavior in environments with high to mild congestion (typical of low speed networks) [5]. In high bandwidth-delay networks in which HighSpeed TCP sends bursts of large number of packets, the amount of buffer available in the bottleneck router is an important issue to keep the router highly utilized during congestion periods. A large buffer increases delay and delay variance which adversely affects real-time applications (e.g., video games, device control and video over IP applications.) It is therefore important to investigate the effects of buffering on TCP performance such as throughput, convergence to fairness and interaction.

## 4. The Model for TCP Congestion Control

Traditionally, experimental study and implementation or measurements have been the tools of choice for examining

the performance of various aspects of TCP. Recently, there is a great interest in analyzing the performance of TCP and TCP-like algorithms to quantify the notion of "TCP-friendliness", which means connections that use non-TCP transport protocol, should get a similar share of bandwidth as TCP connections that share the same link under the same conditions of loss, RTT etc.
In this model, the throughput of TCP's congestion control mechanism is characterized as a function of packet loss and round trip delay. [7]Consider a TCP flow starting at time t=0. For any given time t, define Nt to be the number of packets transmitted in the interval [0, t], and Bt = Nt/t, the throughput on that interval. The steady state TCP throughput B should be:

$$B = \lim_{t\to\infty} B_t = \lim_{t\to\infty} \frac{N_t}{t}$$

A sample path of the evolution of congestion window size is given in figure 1.

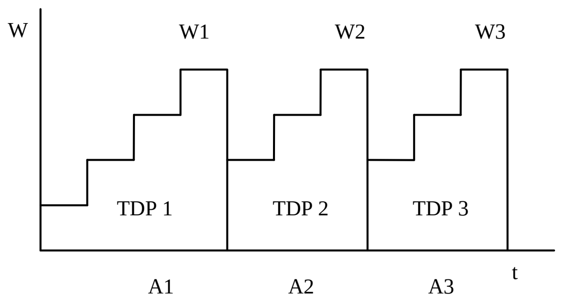

Figure 1 Triple Duplicate Period.

Between two triple duplicate ACKs loss indications, the sender is in congestion avoidance phase. Define a TD period (TDP) to be a period between two triple duplicate ACKs loss indications. For the i-th TDP, define Yi to be the number of packets sent in the period, Ai the duration of the period, and Wi the window size at the end of the period. Considering {Wi}i to be a Markov regenerative process with rewards{Yi}i, it can be shown that

$$B = \frac{E[Y]}{E[A]} \tag{1}$$

It can be derived that

E[Y] = l/p + E[W] - 1
=E[W]*(E[W]*3/2 - 1)*b/4 + E[W]/2 (2)

E[A] =(b*E[W]/2+1)*RTT (3)

Where: p - loss probability, b – window increases by l/b packets per ACK, RTT – round trip time.
From (I) (2) (3), we can get

$$B(p) = \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}}{RTT\left(\frac{2+b}{6}\right) + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1}$$

which can be expressed as:

$$B(p) = \frac{1}{RTT}\sqrt{\frac{3}{2bp} + o\left(\frac{1}{\sqrt{p}}\right)}$$

Extends this model to include the case where the TCP sender times-out and the impact of window limitation of receiver advertised window size, it can be derived that:

$$B(p) = \min\left(\frac{W_{\min}}{RTT^2} - \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_o \min\left(3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)}\right)$$

where $T_o$ - interval waiting for time out, $W_{min}$ - receiver advertised window size.

## 5. Experimental Work

We are using here rate based congestion control protocols, one of which is TFRC (TCP Friendly Rate Control Protocol)[3]. The model assumes that the congestion window size increase is linear during congestion avoidance, which is not always true. When the memory space size of the bottle-neck node is big enough, the increase is sub-linear, which lead to the analytical result will be higher than the actual result. We use QualNet network simulator [6] to verify the assumption.

Table 1: Parameters used in the simulation.

| | |
|---|---|
| Queue Size between R0 & R1 | 40~900 |
| Delay between R0 & R1 | 35 ms |
| Bandwidth between R0 & R1 | 15 Mb |
| TCP window size | 1000 |
| Queue Algorithm | Drop Tail |
| Bandwidth of Branch | 128 Mb |
| Delay of Branch | 2 ms |

We perform 4 experimental tests. Table 2 list the number of connections in each simulation.

Table 2: Number of connections in each simulation.

| ***Experiment No.*** | ***1*** | ***2*** | ***3*** | ***4*** |
|---|---|---|---|---|
| TCP Connections | 16 | 24 | 32 | 64 |
| TFRC Connections | 16 | 24 | 32 | 64 |

In each simulation, we increase the memory space size of RO and RI from 50 to 850 step by 100 and the result is shown in figure 2 - figure 5. In each bar diagram, X axis is memory space size and Y axis is the mean of bandwidth of TCP connections and TFRC connections.

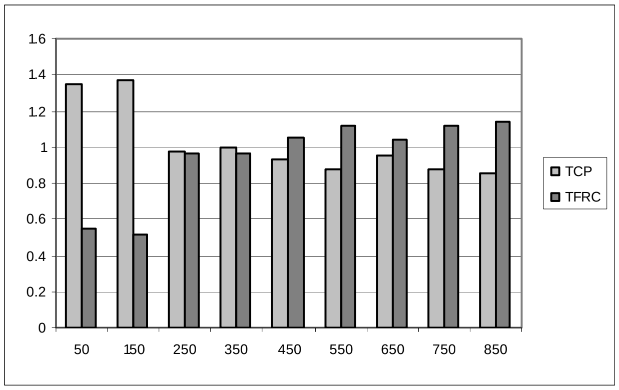

Figure 2 16 TCP & 16 TFRC

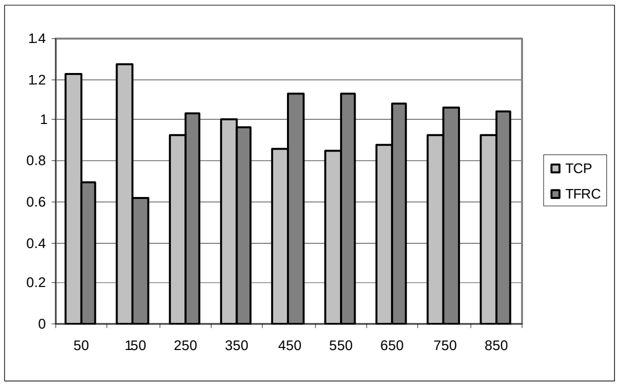

Figure 3 24 TCP & 24 TFRC

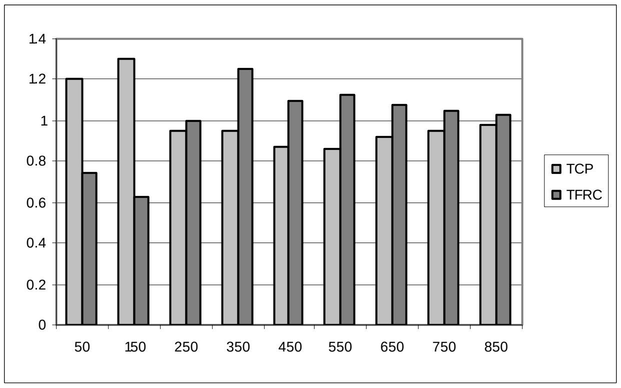

Figure 4 32 TCP & 32 TFRC

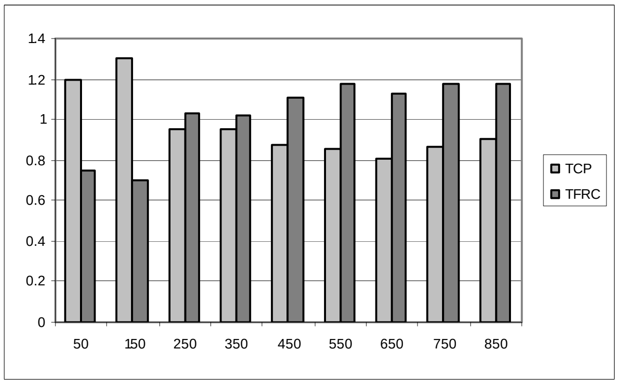

Figure 5 64 TCP & 64 TFRC

## 6. Analysis

From the result of the experiments, we can find that when the memory space size is much less than the Bandwidth*Delay, the model result is quite close to the actual result. However, as the memory space size increases, the experimental study result show that the TFRC connections get more bandwidth than the TCP connections, which means the analytical result, is higher than the actual result. We think it is because when the memory space size is big enough, the congestion window size increase is by sub-linear, not by linear.

Considering equation (3), this implies that the increase of RTT is by linear in a TDP. In fact the increase of RTT is not by linear.

We can find that the RTT will increase much more when the congestion window is big than that when the congestion window is small. When the memory space size of the bottleneck node is small, the congestion window can not be very big, so the RTT can be approximately viewed as increases by linear. But when the memory space size is big enough, the error between linear and quadric curve can not be neglected. The actual time of TDP is longer than the time estimated in the model when the memory space size is big, so the actual throughput is lower than the throughput estimated in the model.

## 7. Related Work

A survey on TCP performance in a heterogeneous network is given in [1]. It considers the different characteristics of a path crossed by TCP traffic, focusing on bandwidth delay product, round trip time (RTT), on congestion losses, and bandwidth asymmetry.
It presents the problems and the different proposed solutions. The model used in this paper was proposed in [7]. TFRC [3] use it to calculate the sending rate of the sender. But only the experimental study result with small memory size was given.

## 8. Conclusion and Future Work

In this paper, we summarize the TCP congestion control mechanism and the analytical model. The model proposed in [7] captures not only the behavior of TCP's fast retransmit mechanism, but also the effect of TCP's timeout mechanism on throughput. But with the big memory space size, the model result does not fit the actual result well. We suppose that it is because the increase of RTT is not by linear when the congestion window is big. We present the experimental study result and give a qualitative analysis. There are a lot of work remain to do. First, a quantitative analysis will help us to make the model more precise. Second, the loss probability of the packet plays an important role in the model. However, in practice, it is hard to measure the loss probability accurately. So, if we get the relation of memory space size and the RTT, we can infer the queue length of the memory space at the bottleneck node from RTT. In our future work, we would also like to investigate the effects of buffer size on the performance of other recently proposed high-speed TCP variants, e.g. FAST TCP [9] which changes its window according to buffer delay.
In this way, we can model the TCP congestion control mechanism mainly based on the queue length of the memory space and the RTT, which can be measured more accurately than packet loss probability.

## References

[l] Chadi Barakat, Eitan Altman, and Walid Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey", IEEE Communications Magazine, January 2000.
[2] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems, Vol 17, pp. 1 - 14, June 1989.
[3] S.Floyd, M.Handley, J.Padhye, and J.Widmer. "Equation-Based Congestion Control for Unicast Applications: the Extended Version", ICSI Technical Report TR-00-03, March 2000.
[4] V. Jacobson, "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.
[5] S. Floyd, "HighSpeed TCP for Large Congestion Windows," in RFC 3649, Experimental, December 2003.
[6] "The Network Simulator—QualNet", Tech. rep., WebPage: http://www.scalable-networks.com Version 5.02, July 2010.
[7] J.Padhye, V.Firoiu, D.Towsley, and J.Kurose, "Modeling TCP throughput: A simple model and its empirical validation", Proceedings of SIGCOMM'98, 1998.
[8] W. Stevens. "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms". RFC 2001, Jan 1997.
[9] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in Proceedings of IEEE INFOCOM'04, March 2004.
[10] Saverio Mascolo and Francesco Vacirca, "TCP Congestion Control & Memory Space Requirements" in Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005 Seville, Spain, December 12-15, 2005.

**Kulvinder Singh** received the M.Tech.(CSE) degree in 2006 and the M.Phil.(CS) degree in 2008 from Ch. Devi Lal University Sirsa(Haryana), India. At present he is working as a Assistant Professor in Vaish College of Engineering, Rohtak, India. He is a member of IEC. He presents many research papers in national and international conferences. His interest areas are Networking, Web Security, Internet Congestion and Fuzzy Database.

**Anil Kumar** received his bachelor degree from Delhi University, Delhi, India and Master degree from IGNOU, India and M.Tech in Computer Science & Eng. From Kurukshetra University, Kurukshetra, India in year 2002 and 2006. Currently he is pursuing Ph.D in Computer Science from the Department of Computer Science & Application – Kurukshetra University, Kurukshetra, India. Currently is Asst. Professor in Computer Science & Engineering Department in Vaish Engineering College, Rohtak, Haryana, India since September, 2006. He had also worked in software industries more than three years & being a lecture in other engineering college more than two years. His research areas include Software engineering, Reengineering, Software Metrics, Object Oriented analysis and design, Reusability, Reliability.