

MODIFICATION OF CONGESTION CONTROL ALGORITHM FOR TCP AND ITS EXTENSION TO EXPLICIT RATE ADJUSTMENT ALGORITHM

by
Angshuman Roy



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

May 2006

MODIFICATION OF CONGESTION CONTROL ALGORITHM FOR TCP AND ITS EXTENSION TO EXPLICIT RATE ADJUSTMENT ALGORITHM

A Thesis Submitted

In Partial Fulfillment of the Requirements

For the Degree of

Master of Technology

by

Angshuman Roy



to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

May 2006

CERTIFICATE

It is certified that the work contained in the thesis entitled “Modification of Congestion Control Algorithm for TCP and Its Extension to Explicit Rate Adjustment Algorithm” by *Angshuman Roy* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

May 2006

(Dr. Y.N.Singh)

Assistant Professor,

Department of Electrical Engineering,

Indian Institute of Technology,

Kanpur-208016

Acknowledgements

This thesis is dedicated to my parents, my brother.

I express my deep sense of gratitude toward my thesis supervisor Dr. Y.N. Singh for his invaluable guidance, moral support and constant encouragement, which helped me to survive through the crests and troughs of my thesis work. It is my pleasure to give my appreciation to him for taking so much interest in my academic and personal welfare.

I would like to my friends specially Soumik, Gopi, and Krishnendu for making my stay at IIT Kanpur a memorable one.

Angshuman Roy

Contents

1	Introduction	1
1.1	Design Goals of MRMCC	3
1.2	Background knowledge related to ERA	5
1.2.1	TCP Friendliness	5
1.2.2	TCP Friendly Rate Estimation	7
1.2.3	Available Bandwidth Estimation using Packet Pair	8
1.2.4	RTT Estimation	10
1.3	Protocol Basics of ERA Algorithm	11
1.4	ERA Algorithms	12
1.4.1	Sender Operation	12
1.4.2	Receiver Operation	13
1.4.3	Rate Adaptation Algorithms	13
1.5	Work Done	16
1.6	Organization of the thesis	16
2	TCP congestion control algorithm and its analytical characteristics	17
2.1	Existing Algorithms	18
2.1.1	Slow Start	18
2.1.2	Congestion Avoidance	19
2.1.3	Fast Retransmission/ Fast Recovery	20
2.2	Analytical characterization of Throughput for TCP Reno algorithm	23
2.2.1	First Model	24

2.2.2	Second Model	25
2.2.2.1	Loss indication by TD	26
2.2.2.2	Loss indication by TD and TO	31
2.3	Modified Algorithm of TCP congestion control	39
2.4	Analytical characteristics of Modified Algorithm	45
3	Theoretical and Simulation Results	50
3.1	Theoretical Comparison	50
3.2	Simulation Results	53
3.3	ERA Improvement	59
4	Conclusion	61
4.1	Future Aspects	63

List of Figures

1	An example of the Internet congestion algorithm	22
2	Evolution of window size over time when loss indications are TD	26
3	Packet sent during a TD period	27
4	Evolution of window size when loss indications are TD & TO	31
5	Packet and ACK transmission preceding a loss indication	34
6	A hypothetical example	38
7	An Ideal model	40
8	Conventional TCP algorithm	41
9	Modified TCP congestion control algorithm	41
10	Modified TCP congestion control algorithm	42
11	Packet sent during an interval	44
12	Theoretical comparisons between conventional TCP & Modified TCP	49
13	Theoretical results of Modified TCP for TD & TO loss	50
14	Theoretical results of Modified TCP for TD loss only	51
15	Throughput comparison for TD loss indication	53
16	Good put comparison for TD loss indication	54
17	Simulation & Theoretical comparison for TD loss indication	55
18	Throughput comparisons for TD & TO loss indication	56
19	Good put comparisons for TD & TO loss indication	57
20	Theoretical & Simulation comparison for TD & TO loss indication	58
21	Performance of Mod. TCP for packet loss due to noise	59
22	Comparison between ERA and improved ERA	61

Chapter 1

Introduction

The Internet's heterogeneity and scale make multipoint communication design a difficult problem. For real-time multimedia, a live signal is broadcast from any particular sender to an arbitrarily large set of receivers along paths with potentially high variability in bandwidth. The simplest solution to this problem is to distribute a uniform representation of the signal to all interested receivers using IP Multicast [1]. Unfortunately, this is suboptimal, because low-capacity regions of the network suffer congestion while high capacity regions are underutilized.

The problems posed by heterogeneity are not just theoretical; they impact on daily use of Internet remote conferencing. For example, a video application is run on a "seminar host" that sources a single rate signal at 128 kbps, the nominal rate for video over the internet Multicast Backbone or Mbone [2]. However, a number of users have high bandwidth connectivity and would prefer to receive higher rate, higher quality video. At the other bandwidth extreme, many users have ISDN access, but a 128 kbps video stream overwhelms an ISDN line. In this open loop approach, the sender broadcasts at some fixed rate without regard to changing network conditions. A better approach is to adjust the transmission rate to match the available capacity in the network, i.e. to react to congestion. Pioneering research in rate adaptive video [3], [4], [5] has shown that this is possible, but unfortunately, in the context of multicast, the notion of network capacity is ill defined. A

control scheme that adjusts the rate of a single stream at the source simply cannot meet the conflicting requirements of a set of heterogeneous receivers. An alternative approach is to combine a layered compression algorithm with a layered transmission scheme [6], [7]. In this approach, a signal is encoded into a number of layers that can be incrementally combined to provide progressive refinement. The relationship among the information contained across the set of layers can either be cumulative or independent. In the cumulative case, each layer provides refinement information to the previous layers and the receiver must subscribe to all layers up to and including the highest layer. In the independent case, each layer is independent and receiver need only subscribe to one layer. This latter scheme is called simulcast because the source transmits multiple copies of the same signal simultaneously at different rates (resulting in different qualities). Definitely, Cumulative scheme is better than simulcast because it makes more effective use of bandwidth.

By dropping layers at choke points in the network, i.e. selectively forwarding only the number of layers that any given link can manage- heterogeneity is managed by locally degrading the quality of the transmitted signal. This framework provides an elegant solution to heterogeneity but system must have mechanisms for determining, communicating, and executing the selective forwarding of sub flows along all the links in a distribution. While much of the previous work leaves this as an implementation detail, a novel mechanism based on IP Multicast was suggested by Deering [8] and elaborated on and/or independently reported in [9], [10], [11], [12], [13]. In this approach, the different layers of the hierarchical signal are striped across multiple multicast groups and receivers adapt to congestion by adding and dropping layers (i.e. joining and leaving multicast

groups). Receivers implicitly define the multicast distribution trees simply by expressing their interest in receiving flows. Thus there is no explicit signaling between the receivers and routers or between the receivers and source. This latter approach to control congestion in Multicast field is called Multirate Multicast congestion control (MR-MCC).

In recent years, several studies have been done on the design of MRMCC protocols [20], [23], [24], [25]. However, all of them have some drawbacks. Some design cause over subscription and high packet losses. Some are slow to converge and unresponsive. Some are TCP unfriendly. Some designs are too complex or even arguable in terms of feasibility. Some others are not scalable. Recently, a new MRMCC protocol named Explicit Rate Adjustment (ERA) has been proposed [14]. According to this algorithm, receiver is able to adjust explicitly its reception rate according to the network conditions using the TCP throughput equation and Packet Pair Probe.

Before discussing about ERA algorithm, it is better to have idea about the design goals of MRMCC algorithm and some background knowledge related to ERA algorithm. These are going to be discussed in next two sections.

1.1 Design Goals of MRMCC

Multi rate Multicast congestion control algorithm should have the following properties:

Scalability: Protocol should be largely scalable such that a sender can send data to a nearly unlimited numbers of receivers.

Responsiveness: Congestion must be detected at the early stage and an immediate action must be taken by receiver by unsubscribing extra layers.

Fast Convergence: Protocol should be designed in such a way that it allows receivers to converge rapidly from any starting state to the stable state with an optimal rate of bandwidth consumption.

Fairness: If more than one connection share same bottleneck bandwidth, then it is desirable that each connection should share it fairly. It is totally unfair that some connections may enjoy a greater share of the bottleneck bandwidth at the expense of other connections. In MRMCC design, following fairness should be considered:

Inter-protocol fairness: When several connections of different protocols compete for bandwidth, they should be able to share it fairly. In particular Internet community suggests that new congestion control mechanism should maintain fairness towards TCP.

Intra-protocol fairness: Fairness among transmission sessions of the same Protocol.

Intra-session fairness: Fairness among receivers of the same multicast session.

High network utilization: Protocol should not under utilize bandwidth if it is available.

Low Packet Loss Rate: Packet loss in internet occurs mainly due to the presence of congestion. As packet loss is waste of bandwidth and it leads to receive poor quality of signal, so packet loss rate should be minimized.

1.2 Background knowledge related to ERA

In this section, some background concepts like TCP-friendliness, TCP-friendly Rate estimation, Available Bandwidth estimation by Packet Pair probe, RTT estimation will be discussed, because all of these are important ingredient of ERA algorithm.

1.2.1 TCP Friendliness

TCP friendliness is actually inter protocol fairness towards TCP. It is defined in [15]. It is a technique for implementing TCP friendly flow control for unicast applications which do not utilize TCP at the transport layer. There are severe consequences to competing unfairly with TCP. Under heavy loads, TCP will back off, reducing its bandwidth utilization. In addition, applications which do not seriously consider congestion issues can contribute to widespread congestive collapse in the Internet. For these reasons it is vitally important that all applications implement some form of congestion control. The basic algorithm incorporated by TCP is Congestion Avoidance. The Congestion Avoidance algorithm probes available network bandwidth by slowly increasing a congestion window. When congestion is detected, TCP reduces the congestion window by half. This rapid back off in response to congestion is the key to TCP's success in avoiding congestive collapse in the Internet. However, this also makes TCP extremely susceptible to bandwidth stealing by other applications which do not implement any control techniques. The widespread development of new, non-TCP based applications poses two major threats to the Internet. First, these applications could contribute to a new congestive collapse of the Internet. Second, these applications will consume an unfairly large portion of resources when run side by side with "good neighbor" TCP applications.

In order to implement a TCP friendly congestion control algorithm, all non-TCP applications (e.g. rate based applications) should simply choose to send at a rate no higher than a TCP connection, operating along the same path, would achieve. Rate based applications are those which do not use a congestion window to control the amount of data outstanding in the network, rather, they choose their sending rate based on what is appropriate for the application. In times when there is no congestion, these applications may send at their desired maximum sending rate. While sending data, the application should monitor for overall packet loss. As long as the overall loss rate is low enough that an equivalent TCP connection would attain at least the same bandwidth, the connection may continue to send at its preferred rate. If the loss rate on the connection rises high enough that an equivalent TCP connection would not be able to attain the same bandwidth, then the rate based application should reduce its bandwidth (as TCP would) by half. The application should then continue to monitor the packet loss on the link. Continued high levels of packet loss may force the applications to perform further reductions in sending rate. If the loss level in the network decreases, the application may increase its sending rate, being careful not to exceed the limits a TCP connection would see under the same loss level.

1.2.2 TCP-friendly Rate Estimation

There have been several analytical and empirical studies to estimate the throughput of TCP in steady state. The first model for TCP throughput has been presented in [15]. From this model, the steady throughput (in bytes per second) of a TCP connection (R_{TCP}) is given as:

$$R_{TCP} = \frac{cM}{RTT * \sqrt{p}} \quad (1.1)$$

Here, RTT is average round trip time and ‘p’ signifies packet loss rate. M is the packet size in bytes and c is a constant which is varying from 0.87 to 1.31.

This model works fine when packet loss is below 5%. For high packet loss rates it over estimates the available bandwidth. In [16], a new model has been proposed which works for a broader range of network conditions. This model gives:

$$R_{TCP} = \frac{M}{RTT * \sqrt{\frac{2bp}{3}} + T_0 \text{Min}\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)} \quad (1.2)$$

Here, b is the number of packets acknowledged by each ACK, T₀ is retransmission time out in seconds. This model is for TCP Reno which is the most widely accepted TCP throughput model by the Internet research community. Both the models will be discussed elaborately in next chapter.

1.2.3 Available Bandwidth Estimation using Packet Pair

To estimate the available bandwidth, it is used the receiver side Packet pair bunches Probe (PP) of Paxson [17] can be used. Before describing about the estimation of available bandwidth it is required to know the difference between Bottleneck bandwidth and Available bandwidth along with the estimation technique of Bottleneck bandwidth. Bottleneck bandwidth gives an upper bound on how fast a connection can possibly transmit data, while Available Bandwidth denotes how fast the connection should transmit

to preserve network stability. Thus, available bandwidth never exceeds bottleneck bandwidth, and can in fact much smaller.

Let ' B_0 ' denotes bottleneck bandwidth of a path and ' t_0 ' denotes the amount of time required to forward a given packet through the bottleneck element. If a packet carries a total of b bytes and the bottleneck bandwidth is B_0 bytes/sec, then, $t_0 = b/B_0$ sec. From a queuing theory perspective, t_0 is simply the service time of a b -bytes packet at the bottleneck link. There will be self interference if sender transmits two b -bytes packets with an interval $t < t_0$ between them, then the second one is guaranteed to have to wait behind the first one at the bottleneck element. The bottleneck estimation technique is based on "packet pair". The fundamental idea is that if two packets are transmitted by the sender with an interval $t < t_0$ between them, then when they arrive at the bottleneck they will be spread out in time by the transmission delay of the first across the bottleneck: after completing transmission through the bottleneck, their spacing will be exactly t_0 . Barring subsequent delay variations, they will then arrive at the receiver spaced not ' t ' apart, but $T_R = t_0$ time apart. So receiver end can estimate the bottleneck bandwidth by $B_0 = b/T_R$.

Now, available bandwidth can also be estimated by similar packet pair technique used for bottleneck bandwidth estimation. If any bottleneck bandwidth is shared by more than one connection and sender of each connection sends data in packet pair form, then for two packets in any packet pair for any connection may not be buffered in bottleneck router back to back, because due to sharing of other connections there is a high probability of buffering some other packets of different connections between these two packets of packet pair. So, second packet of packet pair will not be processed immediately after processing of first packet of that packet pair, rather it will be processed after all packets which are

buffered after the first packet of packet pair. So, in this case time gap between two packets of packet pair at receiver end will be more, and this time gap is proportional to the number of other packets which are buffered between two packets of a packet pair. For any particular connection, receiver can estimate its available bandwidth by receiving more than one packet pairs and estimating corresponding bandwidth for each packet pair by equation $B_0=b/t_0$ and then choosing its available bandwidth as the minimum value of these estimated values.

1.2.4 RTT Estimation

RTT is required for the TCP throughput equation. There are several alternatives proposed to estimate RTT in Multicast. These are described briefly in this section.

Use RTT request packet:

In this technique, receiver send RTT request to sender and sender reply it immediately upon reception. Receiver can estimate RTT by measuring the time difference between sending of RTT request and receiving of the reply packet. However this technique is not suitable for Multicast purpose, because in multicasting sender would be overflowed by RTT requests from large number of receivers. So, some kind of suppression technique must be used to apply this technique to Multicast.

Estimate RTT as twice one-way delay:

The sender transmits a control message every a predefined period with a timestamp to the receivers. When the control message arrives, the receivers estimate half of RTT as the time difference between timestamp and the message arrival time. However one way

delay is not good estimation of half RTT, because this method does not work for asymmetrical paths [18].

Estimate RTT in layered Multicast:

In [19], a new method has been proposed to estimate RTT as the difference between the time of issuance of join request and the arrival time of the packet of the layer. However, this is not exactly the RTT as the join-request messages only propagates back to the router closer to the sender only.

1.3 Protocol Basics of ERA Algorithm

- **Best effort service:** ERA is designed for the Best-effort Service networks. This kind of network does not give any guarantee of quality of service.
- **Single data source:** ERA is one-to-many congestion control protocol. ERA's congestion control is done per source. Multiple data sources can be supported by running multiple instances of ERA.
- **Layered coding and receiver driven:** ERA is designed using the receiver driven layered multicast approach to provide scalability for a very large heterogeneous group of receivers.
- **Explicit rate adjustment:** According to ERA algorithm, the receiver adjusts its reception rate to the target rate, which is explicitly calculated as the minimum of the estimated available bandwidth and the estimated TCP friendly rate.
- **Receiver Coordination:** It is necessary to have receiver coordination under the same bottleneck bandwidth. This coordination is required to obtain maximum

utilization of bandwidth and to control congestion properly. These are revealed in [20]. ERA provides the coordination by relying on Session Announcement Message (SAM) and Rate Adaptation Interval (RAI) as follows. The source sends a SAM at predefined times to provide information about the transmission session. RAI is a part of the information provided in SAM. The receiver can join a transmission session only after it has received a SAM and will adapt its reception rate every RAI. This helps coordinate the subscription and un-subscription among receivers since they adapt their rate more or less at the same time. So, if they are under the same bottleneck link, and traverse the same path, they would have the same target rate estimated from the available bandwidth and the TCP friendly rate.

1.4 ERA Algorithms

In this section ERA algorithm is described. This algorithm is divided into three parts. In first and second part describe the operations required from sender and receiver side respectively and the rate adaptation algorithm is discussed in last part.

1.4.1 Sender Operation

The responsibility of sender is to encode the data into multiple layers. Then, the encoded data packets of each layer are sent as a pair to the receivers. This packet pair will be used to estimate the available bandwidth at the receiver side. The header format of each packet is shown in Table 1. OID identifies which object the packet contains data for. LID identifies which layer the packet is a part of. PSN is used in order to detect packet losses.

SCT indicates the time when the packet is sent from the sender. FPF indicates the first packet of the packet pair. For every predefined Announcing Time (t_{announce}), the sender advertises a Session Announcement Message (SAM) to the receivers. SAM provides a session description with the following information: data rate of each layer, number of layers, IP address of the sender, IP address and port number of each layer, packet size, object length and Rate Adaptation Interval (RAI), which is predefined interval for the receivers to adapt their reception rate.

In ERA algorithm, the layer organization is cumulative. If L_j denotes the data rate of layer j , the cumulative rate (R_i) of a receiver, which subscribes to layer i , can be calculated as:

$$R_i = \sum_{j=0}^i L_j \quad (1.3)$$

Table 1: Packet header format

Name	Description
OID	Object Identifier
LID	Layer Identifier
PSN	Packet Sequence Number
SCT	Sender Current Time

1.4.2 Receiver Operation

The receiver has to receive a SAM and interpret the session description before joining a session. After joining a session, the receiver has a role to decode and obtain the necessary data packets to reproduce the object. Congestion control is done at the receiver side using the algorithm in the next section.

1.4.3 Rate Adaptation Algorithms

- For every arrival of packet pair, the receiver estimates the available bandwidth (R'_{pp}) using the technique mentioned in 1.2.3. If the subscribed rate is higher than R'_{pp} , the receiver will immediately adapt its reception rate down to avoid overloading the network.
- For every RAI, the receiver calculates an estimated bandwidth R_{pp} as the minimum R'_{pp} during the last RAI. There may be a pathological case, when packet pairs are lost during severe congestion. Then, there will not be enough R'_{pp} samples to make a good estimation of available bandwidth. In this case, R_{pp} is set to -1 to indicate severe congestion.
- The receiver also calculates PLR (packet loss rate), Round trip times (RTT), and a TCP friendly Rate using the technique mentioned in 1.2. Let $PLR = p$, and the TCP friendly rate = R_{TCP} .
- The receiver calculates its current subscription rate (R_i) using equation (1.3) with respect to the number of subscribed layers (i) maintained at the receiver, and a data rate of each layer obtained from the session description.
- The receiver estimates the target reception rate (R_{target}) as follows:

```

If (p>0)
  If ( Rpp>=0)
    Set Rtarget = Min ( RTCP , Rpp)
  Else If (Rpp = -1)
    Set Rtarget= RTCP
  Else
    Set Rtarget =Rpp
  End If
End If
End If

```

- The receiver subscribes to or un subscribes from layers according to the R_{target} as follows:

```

If (Ri> Rtarget)
  Repeat Until (Ri <= Rtarget)
    If (i>0)
      Unsubscribe from a layer
      i=i-1
    Else
      Exit the session
    End If
  Loop
Else If (Ri < Rtarget)
  Do While ( Ri+1 < Rtarget)
    Subscribe to a layer
    i= i+1
  Loop
Else
  Maintain the current subscription level
End If
End If

```

1.5 Work Done

In ERA algorithm, target rate is measured by taking minimum of estimated available bandwidth and estimated TCP friendly rate. TCP friendly rate can be estimated by equation (1.2). This equation is derived based on TCP Reno algorithm which can not utilize the available bandwidth completely. In this thesis, a new algorithm is proposed which is modified from TCP Reno algorithm. This modified algorithm can utilize its available bandwidth better than TCP Reno. This improvement is shown by theoretical and simulation comparison between two algorithms in next chapter. If modified TCP algorithm is used for estimating TCP friendly rate, then multicast receivers also can utilize more of its available bandwidth. This thesis also presents an improvement of ERA algorithm by replacing a new equation derived from Modified TCP algorithm for estimating TCP friendly rate.

1.6 Organization of the thesis

Chapter 1 has given the basic background of multicast congestion control algorithm in Internet. Then it discusses the basic goals of multicast congestion control protocol. After that, an efficient congestion control technique named Explicit Rate Adaptation (ERA) algorithm is discussed. Then it is discussed in section 1.5 that how to improve this ERA algorithm in terms of increasing reception rates of multicast receivers.

Chapter 2 describes conventional TCP congestion control algorithm (TCP Reno) which is used all over the Internet for unicast communication and its analytical

characteristics. Then the modification of conventional TCP is presented in terms of utilization of its available bandwidth and analytical characteristics of Modified TCP.

In Chapter 3, Theoretical and simulation comparisons between Conventional TCP and Modified TCP algorithms are shown, and it is also shown that how the implementation of Modified TCP algorithm improves receiving rate of Multicast receivers.

Chapter 4 gives the conclusion and scope for future work.

Chapter 2

TCP Congestion control algorithm and its analytical characteristics

When load offered to any network is more than its capacity, congestion occurs. There are different algorithms that have been developed for controlling congestion in network. Network layer is also able to manage congestion up to certain extent, but real solution to avoid congestion is slow down the data rate, so most of the work is done by TCP. The basic idea is to stop to inject a new packet into the network until an old one is delivered. TCP tries to achieve it by dynamically manipulating the window size [21].

Detecting congestion is the first step to manage it. In past, detecting congestion was difficult. A timeout caused by a lost packet could have been caused by either (1) noise on a transmission line or (2) packet discard at a congested router. But now, packet loss due to transmission errors is relatively rare because most long-haul trunks are fiber. Consequently, most transmission timeouts on the internet are due to congestion. All TCP algorithms assume that timeout are caused by congestion.

When a connection is established, a suitable window size has to be chosen. The receiver can specify a window based on its buffer size. If the sender sticks to this window size, problems will not occur due to buffer overflow at the receiving end, but they may still occur due to internal congestion within the network. So there are two potential problems

exist, (1) network capacity, (2) receiver capacity, and it is necessary to deal with each of them separately. So, each sender maintains two windows: the window the receiver has granted (rwnd) and a second window, the congestion window (cwnd). The effective window is the minimum of the two windows.

2.1 Existing Algorithms

There are different algorithms exist for congestion control in network. But, among these TCP Reno algorithm is most popular implementation of Internet today[22]. So, here only this algorithm will be discussed elaborately. TCP Reno defined four key mechanisms. These are given below:

- (1) Slow start.
- (2) Congestion avoidance.
- (3) Fast retransmission.
- (4) Fast Recovery.

2.1.1 Slow Start

It operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgements are returned by the other end. When a new connection is set up with a host on another network, the congestion window is initialized to one segment. Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit up to the minimum of the congestion window and the advertised window. The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver.

The sender starts by transmitting one segment and waits for its ACK. When ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth. At some point the capacity of the internet can be reached, and an intermediate router will start discarding packets. This informs the sender that its congestion window has become too large.

2.1.2 Congestion avoidance

This algorithm assumes that packet loss caused by damage is very small; therefore the loss of packet indicates congestion somewhere in the network between the source and destination. There are two indications of packet loss: first one is timeout occurring and second is receipt of duplicate ACKs. Although Congestion avoidance and Slow start are independent algorithms with different objectives, but they are implemented together. In the combined algorithm, two variables a congestion window (cwnd) and slow start threshold (ssthresh) are required to be maintained. The algorithm operates as follows:

- Initialization for a given connection sets cwnd to one segment and ssthresh to 64Kb.
- Sender never sends more than the minimum of cwnd and receiver's advertised window.
- When congestion occurs, set ssthresh to one half of the current window size. In addition, if congestion is detected by timeout, set cwnd to one segment (slow start), otherwise, set cwnd to half of the current window size (congestion avoidance).
- When new data is acknowledged by the other end, increase cwnd, but the way of increasing depends on whether TCP is performing slow start or Congestion avoidance.

If cwnd is less than ssthresh, TCP is in slow start; otherwise TCP is in congestion avoidance. Slow start continues until TCP is reached to ssthresh, and then congestion

avoidance starts. In Slow start cwnd is increased exponentially but in Congestion avoidance cwnd is increased by $(\text{segsz} \times \text{segsz} / \text{cwnd})$ each time an ACK is received, where segsz is the segment size and cwnd is maintained in bytes i.e. in congestion avoidance cwnd increases linearly. The increase in cwnd should be at most one segment each round trip time (regardless how many ACKs are received in that RTT), whereas slow start increments cwnd by the number of ACKs received in a round trip time.

2.1.3 Fast Retransmission/ Fast Recovery

When an out of order segment arrives TCP receiver should send an immediate duplicate ACK. 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 point of view, duplicate ACK s 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 ACK s. Second, duplicate ACK s can be caused by the re-ordering of 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.

The TCP sender should use the "fast retransmit" algorithm to detect and repair loss, based on incoming duplicate ACK s. The fast retransmit algorithm uses the arrival of three duplicate ACK s as an indication that a segment has been lost. After receiving three duplicate ACK s, TCP performs retransmission of missing segment, without waiting for the retransmission timer to be expired.

After the fast retransmission algorithm sends what appears to be the missing segment, the "fast recovery" algorithm governs the transmission of new data until a non-duplicate ACK arrives. The reason for not performing slow start in this case is that the

receipt of the duplicate ACK s tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment received, that segment has left the network and is in receiver's buffer, i.e. there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

The fast retransmit and fast recovery algorithms are implemented together as follows:

- (1) When the third duplicate ACK is received, set $ssthresh$ to one half the current congestion window, $cwnd$, but not less than two segments.
- (2) Retransmit the lost segment and set $cwnd$ to $ssthresh$ plus $3 \times segsize$. This artificially inflates the congestion window by the number of segments (three) that have left the network and which the receiver buffered.
- (3) For each additional duplicate ACK received, increment $cwnd$ by $segsize$. This artificially inflates the congestion window in order to reflect the additional segment that has left the network.
- (4) Transmit a segment, if allowed by the new value of $cwnd$ and the receiver's advertised window.
- (5) When the next ACK arrives that acknowledges new data, set $cwnd$ to $ssthresh$ (the value set in step (1)).

This ACK should be the acknowledgement of the retransmission from step (1), one round trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one half the rate it has at when the packet has lost.

This algorithm is shown in Figure 1. In this figure a connection is established by initializing congestion window (cwnd) to one Kilo bytes (a full segment size) and ssthresh to 30 Kilo bytes. Initially TCP sender adapts slow start algorithm to trace the network condition and it is continued till cwnd reached to ssthresh, after that congestion algorithm takes over. In this figure first congestion is detected by receiving triple duplicate acknowledgement by sender. At this moment congestion window size is say 40 Kbytes. So, TCP sender sets its ssthresh and cwnd to 20 Kbytes (half of the current window size) and starts congestion avoidance algorithm. It is shown in this figure that next congestion is detected by time out and cwnd size at this moment is say 50 Kbytes. So, sender sets its ssthresh to 25 Kbytes (half of the current congestion window size) and cwnd to 1 Kbytes (a full segment), and then it starts slow start algorithm again.

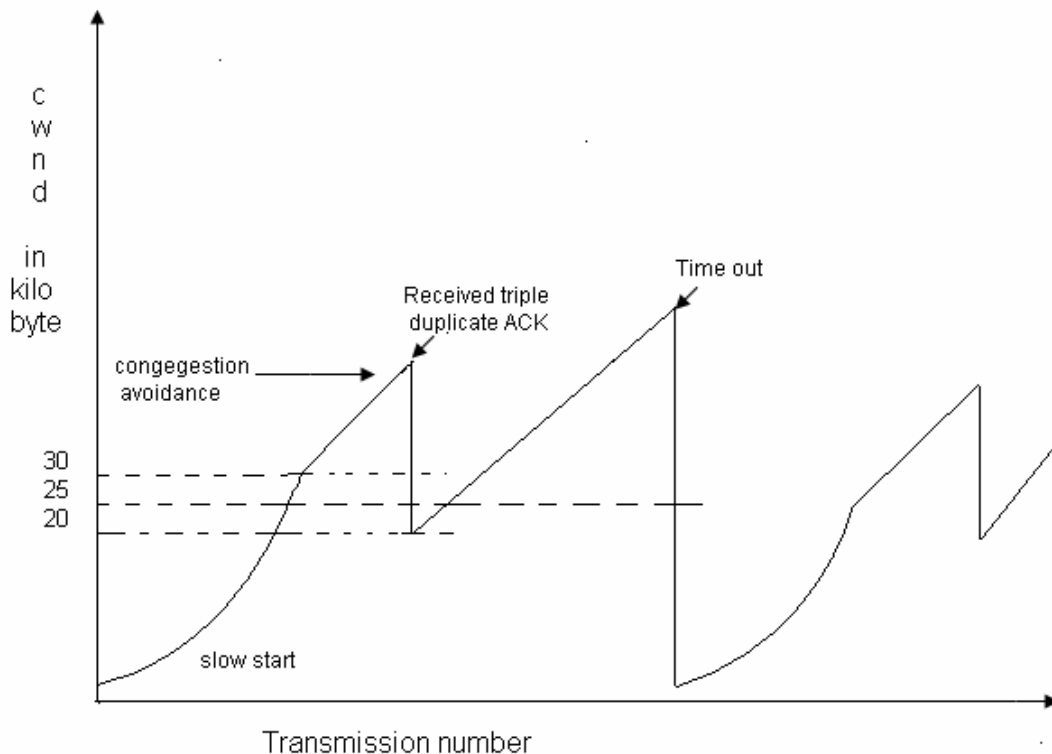


Fig 1: An example of the Internet congestion algorithm

2.2 Analytical Characterization of throughput for TCP Reno algorithm

It is necessary to find out a technique for implementing TCP friendly flow control for unicast applications which do not utilize TCP at the transport layer. There are severe consequences to competing unfairly with TCP. Under heavy loads, TCP will back off reducing its bandwidth utilization. In addition, applications which do not seriously consider congestion issues can contribute to widespread congestive collapse in the Internet. For these reason it is vitally important that all applications implement some form of congestion control. Recently, however several efforts have been directed at analytically characterizing the throughput of TCP's congestion control mechanism as a function of packet loss and round trip delay. One reason for this interest is that a simple quantitative characterization of TCP throughput under given operating conditions offers the possibility of defining a "TCP friendly" throughput for a non-TCP flow that interacts with a TCP connection. Here, two type of model will be discussed. In "First Model" TCP throughput equation is simple one, but it is applicable for low packet loss(less than 5%)[15]. For high packet loss this equation overestimates the bandwidth. But in "Second Model" it is possible to predict accurately the throughput over a significantly wider range of loss rates than "First Model"[16].

2.2.1 First Model

Consider a TCP connection with a particular round trip time and packet size. Further consider a steady state model where the network drops a packet from that connection when the connection's congestion size increases to W packets. Assume that the congestion window is then cut in half, and then is increased by one packet per round trip time until it reaches W packets again, at which point the network again drops a packet and the steady state model continues as before.

Assume that the TCP connection has MTU bytes/packet and a round trip time of RTT seconds. Assume that, when a packet is dropped, the TCP connection had a window of W packets, and was sending at an average rate (over that round trip time) of,

$$S = W \times MTU / RTT \quad \text{bytes/second}$$

After the packet is dropped, it takes roughly $W/2$ roundtrip times for the TCP sender's congestion window to build up again until it reaches its old value. Thus, in steady state the TCP connection receives an average bandwidth of $0.75 \times S$ bytes/sec, because sending rate varies smoothly between $S/2$ and S bytes/second.

$$\text{So, the average bandwidth} = 0.75 \times W \times MTU / RTT \quad (2.1)$$

Now, the loss rate for that TCP connection, is

$$\text{Loss} = 1 / (W/2 + (W/2 + 1) + \dots + W)$$

$$\text{Loss} \sim 1 / ((3/8) W^2).$$

$$\text{So, } W = \sqrt{8 / (3 \times \text{Loss})}.$$

Substituting for W in eq (2.1), this gives,

$$\text{Bandwidth} = 1.22 \times MTU / (RTT \times \sqrt{\text{Loss}}).$$

2.2.2 Second Model

In this model it is derived a simple analytical expression for the throughput of a saturated TCP sender i.e. a flow with an unlimited amount of data to send as a function of loss and average round trip time. In first model only fast retransmit mechanism is considered, but experimentally it is found that time out event occurs more frequent than fast retransmit event in almost all TCP connections. So, in this model time out event is also considered. This model is described in [16]. It focuses on TCP congestion avoidance mechanism. Congestion avoidance mechanism is characterized in terms of “rounds”. A round starts with back-to back transmission of W packets, where W is the current size of the TCP congestion window. Sender refrains to transmit any packet after sending all packets in congestion window until the first ACK received for one of these W packets. This ACK reception indicates the end of the current round and beginning of the next round. In this model it is also assumed that the time needed to send all packets in a window is smaller than the round trip time. So, the duration of a round is equal to the round trip time and is also independent of the window size.

If in current round, W packets are sent then in next round $W' = W + 1/b$ packets will be sent in absence of loss, where b is the number of packets that are acknowledged by a received ACK. In this model an important assumption is taken. This assumption is that a packet is lost in a round independently of any packet lost in other rounds and packet losses are correlated among the back to back transmission within a round, i.e. if a packet is lost, all remaining packets transmitted until the end of that round are also lost[16].

In the next section two different cases are considered depends on the packet losses detected by receiving triple duplicate ACK (TD) only and by time out (TO) and TD both.

2.2.2.1 Loss indication by TD

Here it is assumed that packet losses are detected by receiving triple duplicate ACKs (TD) only and it is also assumed that receiver has a large amount of buffer to store packets, so sender's window size is not restricted by receiver's advertised flow control. Suppose N_t is number of packet transmitted in an interval of duration t , then the throughput on that interval is $B_t = N_t/t$. Since B_t is the number of packets sent per unit of time regardless of their eventual fate, B_t represents the throughput of the connection, rather than its goodput. The long term steady state TCP throughput B is defined to,

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

It is assumed that if a packet is lost in a round, all remaining packets transmitted until the ends of the round are also lost. Therefore let's define p to be the probability that a packet is lost, given that either it is the first packet in its round or the preceding packet in its round is not lost. A sample path of the evolution of congestion window size is given in Figure 2.

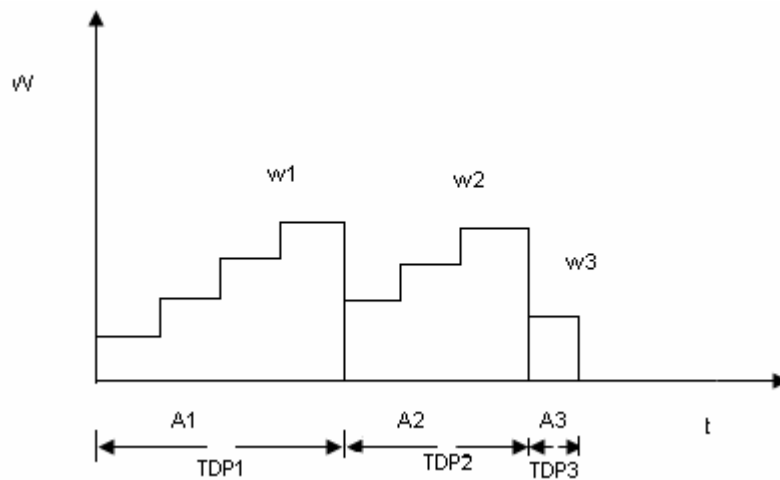


Fig 2: Evolution of window size over time when loss indications are TD

Between two TD loss indications the sender is in congestion avoidance, and the window is increased by $1/b$ packets per round. Immediately after the loss indication occurs, the window size is reduced by a factor of two. It is defined a TD period (TDP) to be a period, between two TD loss indications. For the i -th TD period let's assume $Y(i)$ to be the number of packets sent in the period. A_i the duration of the period, and $W(i)$ the window size at the end of the period. Considering $\{W(i)\}_i$ to be a Markov regenerative process with rewards $\{Y(i)\}_i$, it can be shown that,

$$B = \frac{E[Y]}{E[A]} \quad (2.2).$$

In order to derive an expression for B , the long term steady state TCP throughput, it is necessary to derive expression for the mean of Y and A .

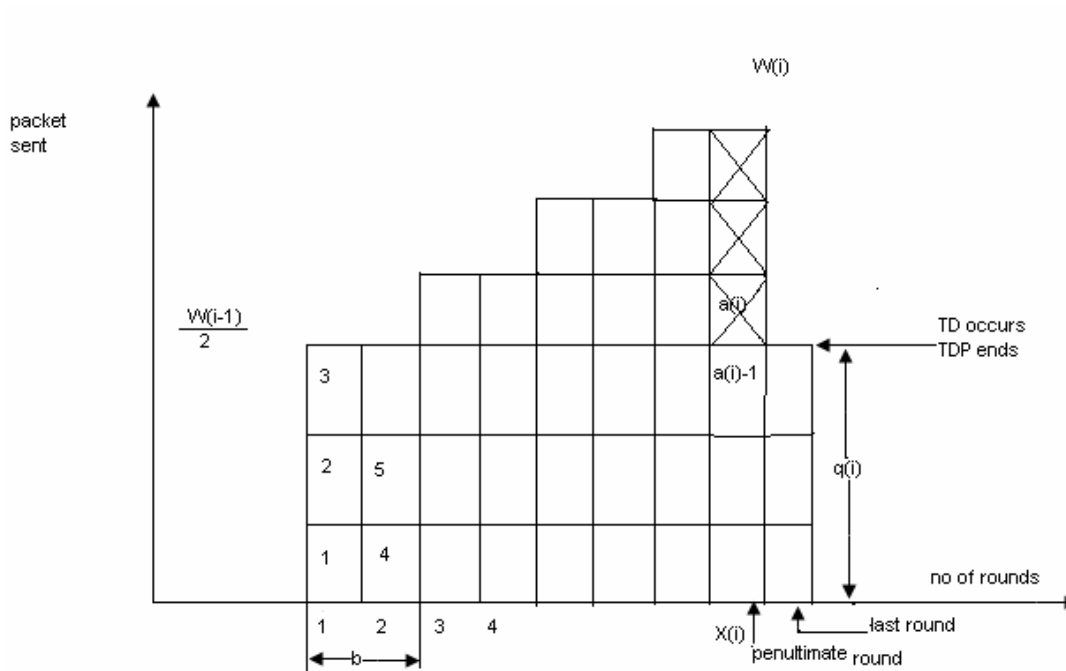


Fig 3: Packet sent during a TD period

Consider a TD period as in Figure 3. A TD period starts immediately after a TD loss indication, and thus the current congestion window size is equal to $W(i-1)/2$, half the size of window before the TD occurred. At each round the window is incremented by $1/b$ and the number of packets sent per round is incremented by one in every b rounds. Let's denote $a(i)$ the first packet lost in TDP_i , and $X(i)$ the round where this loss occurs. After packet $a(i)$, $W(i) - 1$ more packets are sent in an additional round before a TD loss indication occurs. This is described in more details in next section. Thus, a total of $Y(i) = a(i) + W(i) - 1$ packets are sent in $X(i) + 1$ rounds. It follows that:

$$E[Y] = E[a] + E[W] - 1 \quad (2.3)$$

To derive $E[a]$, consider the random process $\{a(i)\}_i$, where $a(i)$ is the number of packets sent in TD period up to and including the first packet that is lost. Based on the assumption that packets are lost in a round independently of any packets lost in other rounds, $\{a(i)\}_i$ is a sequence of independent and identically distributed random variables. The probability that $a(i)=k$ is equal to the probability that exactly $k-1$ packets are successfully acknowledged before a loss occurs,

$$P[a = k] = (1-p)^{k-1} p, \quad k=1,2,3,4... \quad (2.4)$$

The mean of "a" is thus

$$E[a] = \sum_{k=1}^{\infty} (1-p)^{k-1} pk = \frac{1}{p} \quad (2.5)$$

From (2.3) and (2.5) ,

$$E[Y] = \frac{1-p}{p} + E[W] \quad (2.6)$$

To derive $E[W]$ and $E[A]$, consider again TDP_i . Let's denote $r(i, j)$ to be the duration of the j -th round of TDP_i .

Then, the duration of TDP_i is,

$$A(i) = \sum_{j=1}^{X(i)+1} r(i, j)$$

It is considered that the round trip times $r(i, j)$ are random variables, that are assumed to be independent of size of congestion window, and thus independent of the round number, j . It follows that,

$$E[A] = (E[X] + 1)E[r] \quad (2.7)$$

From now, average value of round trip time $E[r]$ will be denoted by RTT.

For simplification it is assumed that $W(i-1)/2$ and $X(i)/b$ are integers. It can be observed in Figure (3) that during i -th period, the window size increases between $W(i-1)/2$ and $W(i)$. Since the increase is linear with slope $1/b$, we have:

$$W(i) = \frac{W(i-1)}{2} + \frac{X(i)}{b}, \quad i=1,2,3,4\dots \quad (2.8)$$

The fact that $Y(i)$ packets are transmitted in TDP_i is expressed by,

$$Y(i) = \sum_{k=0}^{(X(i)/b)-1} \left(\frac{W(i-1)}{2} + k \right) b + q(i) \quad (2.9)$$

$$= \frac{X(i)W(i-1)}{2} + \frac{X(i)}{2} \left(\frac{X(i)}{b} - 1 \right) + q(i) \quad (2.10)$$

Using equation (2.8),

$$Y(i) = \frac{X(i)}{2} \left(\frac{W(i-1)}{2} + W(i) - 1 \right) + q(i) \quad (2.11)$$

Where $q(i)$ is the number of packets sent in the last round. $\{W(i)\}_i$ is a Markov process for which a stationary distribution can be obtained numerically, based on (2.8) and (2.11) and

on the probability density function of $\{a(i)\}$ given in (2.4). It can also be computed the probability distribution of $\{X(i)\}$. A simpler approximate solution can be obtained by assuming that $\{X(i)\}$ and $\{W(i)\}$ are mutually independent sequences of i.i.d random variables. With this assumption, it follows from (2.8), (2.11) and (2.6) that

$$E[W] = \frac{2}{b} E[X] \quad (2.12)$$

and,
$$\frac{1-p}{p} + E[W] = \frac{E[X]}{2} \left(\frac{E[W]}{2} + E[W] - 1 \right) + E[q] \quad (2.13)$$

It is considered that $q(i)$, the number of packets in the last round, is uniformly distributed between 1 and $W(i)$, and thus $E[q] = E[W]/2$. From (2.12) and (2.13), we have,

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \quad (2.14)$$

From (2.12), (2.7) and (2.14), it follows,

$$E[X] = \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} \quad (2.15)$$

$$E[A] = RTT \left(\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right) \quad (2.16)$$

From (2.2) and (2.6) we have,

$$B(p) = \frac{\frac{1-p}{p} + E[W]}{E[A]} \quad (2.17)$$

$$B(p) = \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}}{RTT \left(\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1 \right)} \quad (2.18)$$

2.2.2.2 Loss indication by TD and TO

So far, it is considered TCP flows where all loss indications are due to “triple-duplicate” ACKs. It is found in many cases the majority of window decreases are due to time-outs, rather than fast retransmits. Therefore, a good model should capture time-out loss indications. In this section TO loss is also included. Sender time out occurs when packets (or ACKs) are lost, and less than three duplicate ACKs are received. The sender waits for a period of time denoted by T_0 , and then retransmits non-acknowledged packets. Following a time-out, the congestion window is reduced to one, and one packet is thus resent in the first round after a time out. In the case that another time-out occurs before successfully retransmitting the packets lost during the first time out, the period of time out doubles to $2T_0$; this doubling is repeated for each unsuccessful retransmission until $64T_0$ is reached, after which the time out period remains constant at $64T_0$.

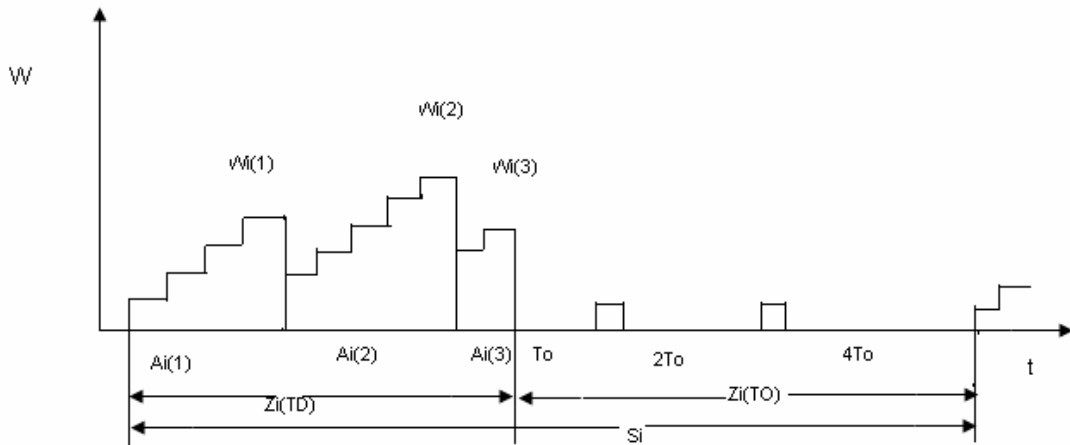


Fig 4: Evolution of window size when loss indications are TD & TO

An example of the evolution of congestion window size is given in Figure 4. Let $Z_i(\text{TO})$ denote the duration of sequence of time-outs and $Z_i(\text{TD})$ the time interval between two consecutive time out sequences. Define S_i to be,

$$S_i = Z_i(\text{TD}) + Z_i(\text{TO})$$

Also, define M_i to be the number of packets sent during S_i . Then, $\{(S_i, M_i)\}$ is an i.i.d sequence of random variables, and we have

$$B = \frac{E[M]}{E[S]}$$

Here, the definition of TD periods given in last section is extended to include periods starting after, or ending in a TO loss indications (besides periods between two TD loss indications). Let $n(i)$ be the number of TD periods in interval $Z_i(\text{TD})$. For the j -th TD period of interval $Z_i(\text{TD})$ lets define $Y(i,j)$ to be the number of packets sent in the period, $A(i,j)$ to be the duration of period, $X(i,j)$ to be the number of rounds in the period, and $W(i,j)$ to be the window size at the end of the period. Also, $R(i)$ denotes the number of packets sent during time-out sequence $Z_i(\text{TO})$. Here $R(i)$ counts the total number of packet transmissions in $Z_i(\text{TO})$, and not just the number of different packet sent. This is because, as discussed in last section, here throughput of TCP flow is calculated not its goodput. We have,

$$M_i = \sum_{j=1}^{n(i)} Y(i, j) + R(i) \qquad S_i = \sum_{j=1}^{n(i)} A(i, j) + Z_i(\text{TO})$$

And thus,

$$E[M] = E\left[\sum_{j=1}^{n(i)} Y(i, j)\right] + E[R] \qquad E[S] = E\left[\sum_{j=1}^{n(i)} A(i, j)\right] + E[Z(TO)]$$

If it is assumed that $\{n(i)\}_i$ is an i.i.d sequence of random variables, independent of $\{Y(i,j)\}$ and $\{A(i,j)\}$, then ,

$$E\left[\left(\sum_{j=1}^{n(i)} Y(i, j)\right)_i\right] = E[n]E[Y] \qquad E\left[\left(\sum_{j=1}^{n(i)} A(i, j)\right)_i\right] = E[n]E[A]$$

To derive $E[n]$ observe that, during $Z_i(\text{TD})$, the time between two consecutive time-out sequences, there are $n(i)$ TDPs where each of the first $n(i)-1$ end in a TD, and the last TDP ends in a TO. It follows that in $Z_i(\text{TD})$ there is one TO out of $n(i)$ loss indications. Therefore, the probability (Q) that a loss indications ending a TDP is a TO, is

$$Q = \frac{1}{E[n]}$$

Consequently,

$$B = \frac{E[Y] + Q \times E[R]}{E[A] + Q \times E[Z(TO)]} \qquad (2.19)$$

Since $Y(i,j)$ and $A(i,j)$ do not depend on time-outs, their means are those derived in(2.5) and (2.16). To compute TCP throughput using (2.19) it is necessary to determine Q , $E[R]$ and $E[Z(TO)]$.

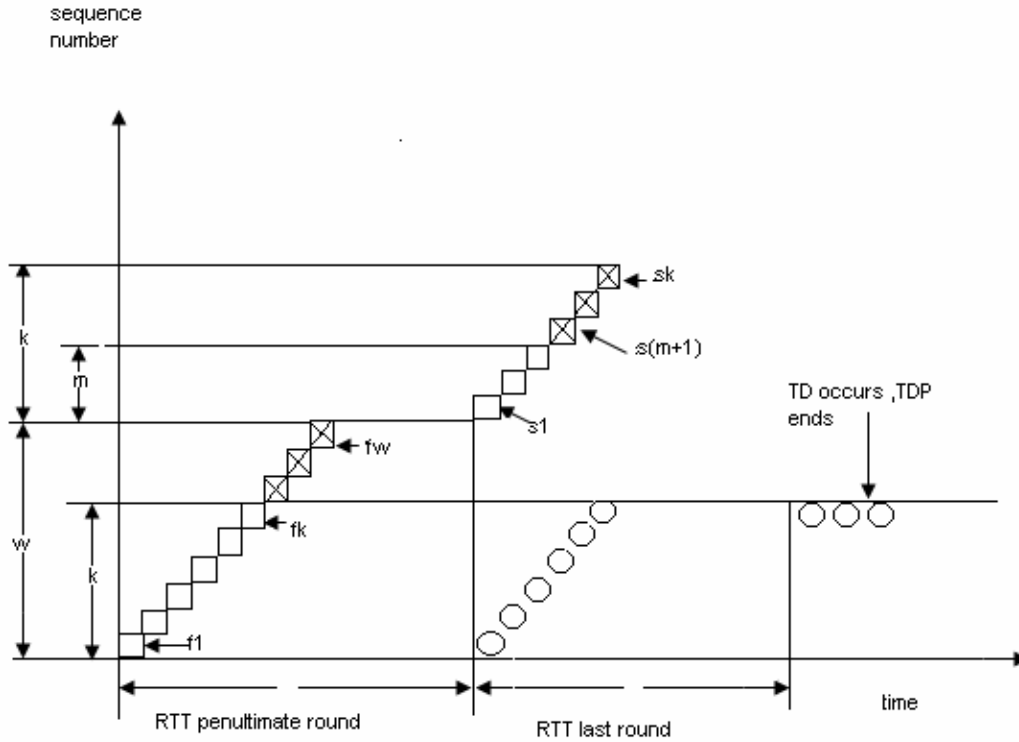


Fig 5: Packet and ACK transmission preceding a loss indication

Let's begin by deriving an expression for Q . Consider the round of packets where a loss indication occurs, it will be referred to as the "penultimate" round (in Fig 5). Let w be current congestion window size. Thus packets f_1, \dots, f_w are sent in the penultimate round. Packets f_1, \dots, f_k are acknowledged, and packet f_{k+1} is the first one to be lost. It is assumed that packet losses are correlated within a round: if a packet is lost, so are all packets that follow, till the end of the round. Thus, all packets following f_{k+1} in the penultimate round are also lost. However, since packets f_1, \dots, f_k are ACKed, another k packets, s_1, \dots, s_k are sent in the next round, which is referred to the "last" round. This round of packets may have another loss, say packet s_{m+1} . So packets s_{m+2}, \dots, s_k are also lost in the last round. The m packets successfully sent in the last round are responded to by ACKs for packet f_k , which are counted as duplicate ACKs. These ACKs are not delayed, so the number of

duplicate ACKs is equal to the number of successfully received packets in the last round. If the number of such ACKs is higher than three, then a TD indication occurs, otherwise, a TO occurs. In both cases the current period between losses, TDP, ends. If $A(w,k)$ is the probability that the first k packets are ACKed in a round of w packets, given there is a sequence of one or more losses in the round. Then,

$$A(w,k) = \frac{(1-p)^k p}{1-(1-p)^w}$$

If $C(n,m)$ to be the probability that m packets are ACKed in sequence in the last round and the rest of the packets in the round, if any, are lost. Then,

$$\begin{aligned} C(n,m) &= (1-p)^m p, \quad m \leq n-1 \\ &= (1-p)^n, \quad m=n \end{aligned}$$

Then $Q(w)$, the probability that the loss in a window of size is a TO, is given by,

$$\begin{aligned} Q(w) &= 1 && w \leq 3 \\ &= \sum_{k=0}^2 A(w,k) + \sum_{k=3}^w A(w,k) \sum_{m=0}^2 C(k,m) && \text{Otherwise} \end{aligned} \quad (2.20)$$

Since a TO occurs if the number of packets successfully transmitted in the penultimate round, k , is less than three or otherwise if the number of packets successfully transmitted in the last round, m is less than three. Also, due to the assumption that packet s_{m+1} is lost independently of packet f_{k+1} (since they occur in different rounds), the probability that there is a loss at f_{k+1} in the penultimate round and a loss at s_{m+1} in the last round equals $A(w,k) \times C(k,m)$, and (2.19) follows.

After algebraic manipulations, it can be found,

$$Q(w) = \min\left(1, \frac{(1 - (1 - p)^3)(1 + (1 - p)^3(1 - (1 - p)^{w-3}))}{1 - (1 - p)^w}\right) \quad (2.21)$$

Using L'Hopital's rule,

$$\lim_{p \rightarrow \infty} Q(w) = \frac{3}{w}$$

Numerically it is found that a very good approximation of Q is,

$$Q(w) \approx \min\left(1, \frac{3}{w}\right) \quad (2.22)$$

Q' the probability that a loss indication is a TO, is

$$Q' = \sum_{w=1}^{\infty} Q(w)P[W = w] = E[Q]$$

It is approximated, $Q' \approx Q(E[W])$ (2.23)

where E[W] is from (2.14).

For the derivation of E[R] and E[Z(TO)], we need the probability distribution of the number of timeouts in a TO sequence, given that there is a TO. It is observed in TCP traces that in most cases one packet is transmitted between two time-outs in sequence. Thus, a sequence of k TOs occurs when there are k-1 consecutive losses (the first loss is given) followed by a successfully transmitted packet. Consequently, the number of TOs in a TO sequence has a geometric distribution, and thus

$$P[R=k] = p^{k-1}(1-p)$$

Then we can compute R's mean

$$E[R] = \sum_{k=1}^{\infty} kP[R = k] = \frac{1}{1-p} \quad (2.24)$$

For the calculation of $E[Z(TO)]$, the average duration of a time-out sequence retransmissions, which can be computed in a similar way. We now that first six time-outs in one sequence have length $2^i T_0$, $i=1,2,\dots,6$, with all immediately following timeouts having length $64T_0$. Then, the duration of a sequence with k time-outs is,

$$\begin{aligned} L_k &= (2^k - 1)T_0 && \text{for } k \leq 6 \\ &= (63 + 64(k-6)T_0 && \text{for } k \geq 7 \end{aligned}$$

And the mean of $Z(TO)$ is,

$$\begin{aligned} E[Z(TO)] &= \sum_{k=1}^{\infty} L_k P[R = k] \\ &= T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1 - p} \end{aligned}$$

Armed now with expressions for Q , $E[S]$, $E[R]$ and $E[Z(TO)]$ we can now substitute these expressions into equation(2.19) to obtain the following for $B(p)$:

$$B(p) = \frac{\frac{1-p}{p} + E[W] + Q(E[W]) \frac{1}{1-p}}{RTT(E[X] + 1) + Q(E[W])T_0 \frac{f(p)}{1-p}} \quad (2.25)$$

where, $f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$

Q is given in(2.21), $E[W]$ in (2.14) and $E[X]$ in (2.15). Using (2.22),(2.14) and (2.15), eq(2.25) can be approximated by,

$$B(p) \approx \frac{1}{RTT \sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)} \quad (2.26)$$

2.3 Modified Algorithm of TCP congestion control

In last section, it was discussed about a conventional TCP congestion control algorithm. This algorithm works fine in almost all kind of congestion in network. But, does this algorithm allow a connection to utilize its total available bandwidth? Let's see it with an example. Consider a hypothetical case where available bandwidth for a connection is almost constant in all over its transfer time.

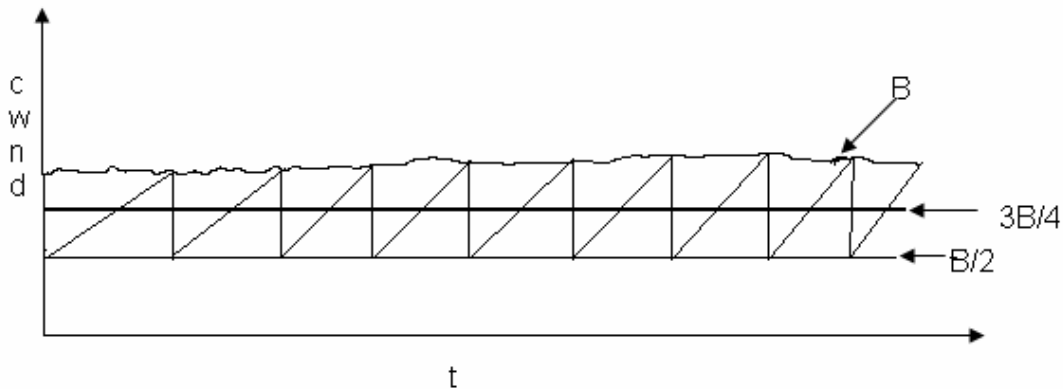


Fig 6: A hypothetical example

It is shown in Figure (6). Suppose this constant rate is B bytes/sec. Sender of this connection runs the conventional TCP algorithm in its side. For simplicity we assume that sender increases its rate linearly (congestion avoidance) for every successful transmission and decrease its rate by factor of two when packet losses are detected. From Figure (1) we can see that sender increase its rate from $B/2$ to B linearly and after that it detects congestion and it slow down its rate to $B/2$ and then it starts all over again. Therefore the average transmission rate for this connection is between $B/2$ and B , i.e. $(B + B/2) / 2 = 3B/4$ bytes/sec.

i.e. Bandwidth utilization = $\frac{3B/4}{B} \times 100\% = 75\%$.

So TCP connection uses only 75% of its available bandwidth. It will be lower if sender includes slow start and Time out (which occurs frequently) in this algorithm. So, is there any alternative algorithm that gives better bandwidth utilization for TCP connection? Yes, it can be possible if sender is able to know by somehow about the condition of network, and then adapt its rate based on this information. Sender receives the information of network condition from receiver. Receiver can estimate the bandwidth of network by using packet pair probe technique. In this technique sender should send its data in packet pair form. Two packets in packet pair has to be send in back to back. They will reach at receiver end with a significant time difference between them, which is proportional to the bottleneck bandwidth of the connection. If this time difference is t_0 and size of the packet is b bytes then estimated bandwidth is , $BW= b/t_0$ bytes/sec.

In this way receiver can estimate the bandwidth after receiving each packet pair probe. Now, what receiver has to do, is to observe the network for a predefined time interval and it has to estimate the available bandwidth R' for every arrival of packet pair and sends the estimated bandwidth R as the minimum R' during this interval. After getting this feedback, sender modifies its congestion window and sticks to this window until it sees any congestion in network and then immediately decreases its window size based on what kind of congestion it observed i.e. sender time out or receiving triple duplicate ACKs. After that sender starts its conventional TCP congestion algorithm till the end of interval and then modify its congestion window based on feedback received from receiver side and starts all over again.

Now, let's see how this new algorithm improves the bandwidth utilization factor. Before that we have to know what should be the idle model that allows a connection to

utilize maximum of its available bandwidth. Let's consider the Figure (7). In this figure thin line represents the maximum available bandwidth for a TCP connection. If sender can adjust its congestion window according to available bandwidth which is shown in Figure (7) by bold line then it can be possible to utilize maximum of its available bandwidth. Now let's see that what amount of bandwidth conventional and modified TCP algorithm can utilize for the same network conditions.

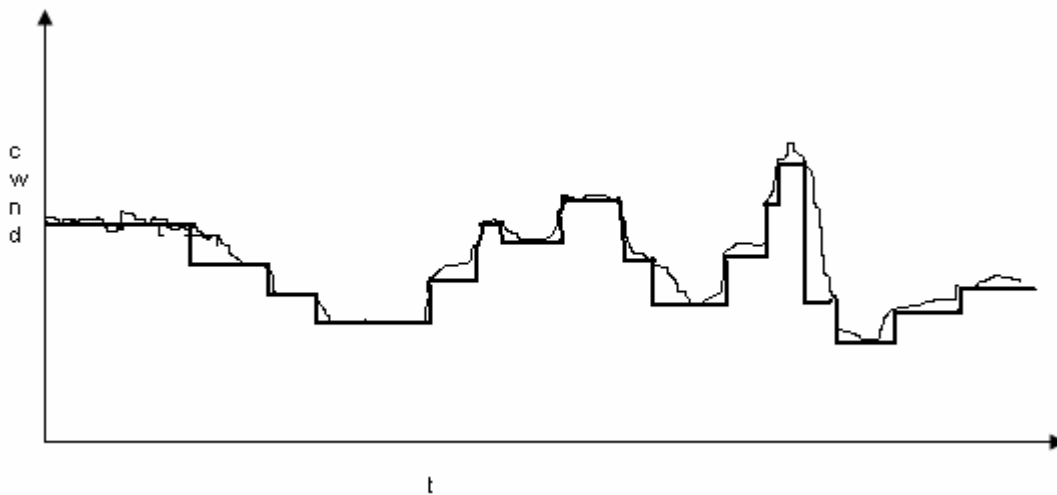


Fig 7: An Ideal model

It is shown in Figure (8) and Figure (9) respectively. From Figure(8) we can see that a significant amount of available bandwidth is wasted for conventional TCP algorithm where as TCP connection can use more bandwidth with modified algorithm which is shown in Figure(9). Here, T denotes the duration of interval during which receiver observe the network and send feedback at the end of the interval. Now if we decrease this duration of interval by factor of two, i.e. to allow receiver to observe the network during an interval of duration $T/2$ and send feedback after each interval, then TCP connection can use more of its available bandwidth, which is shown in Figure (10).

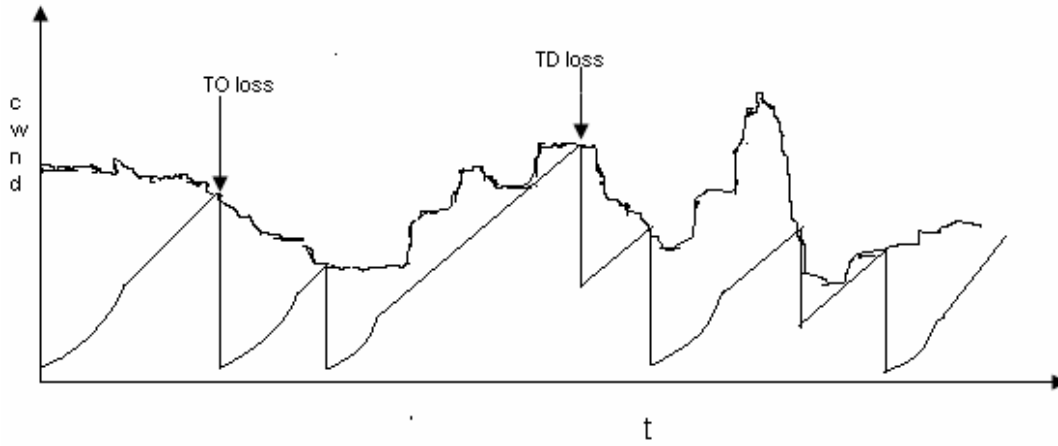


Fig 8: Conventional TCP algorithm

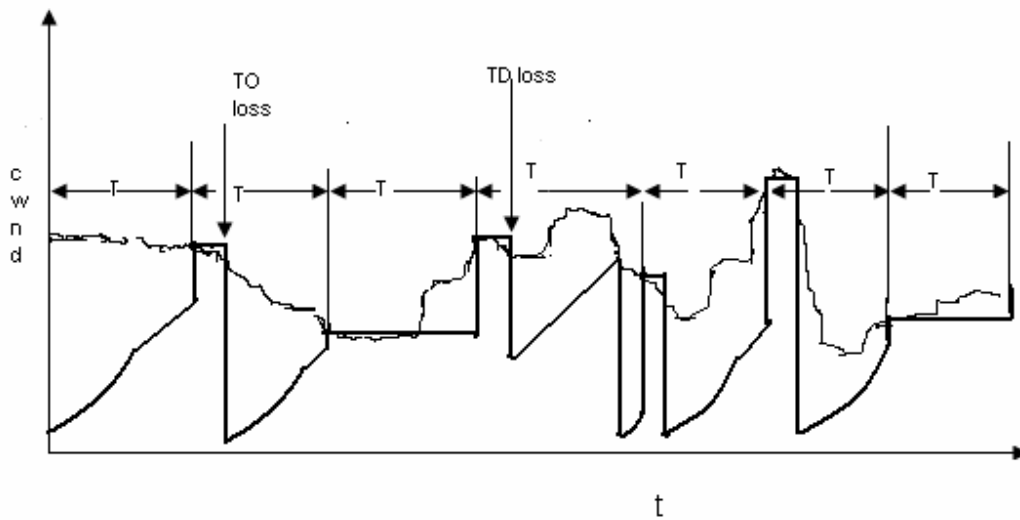


Fig 9: Modified TCP congestion control algorithm

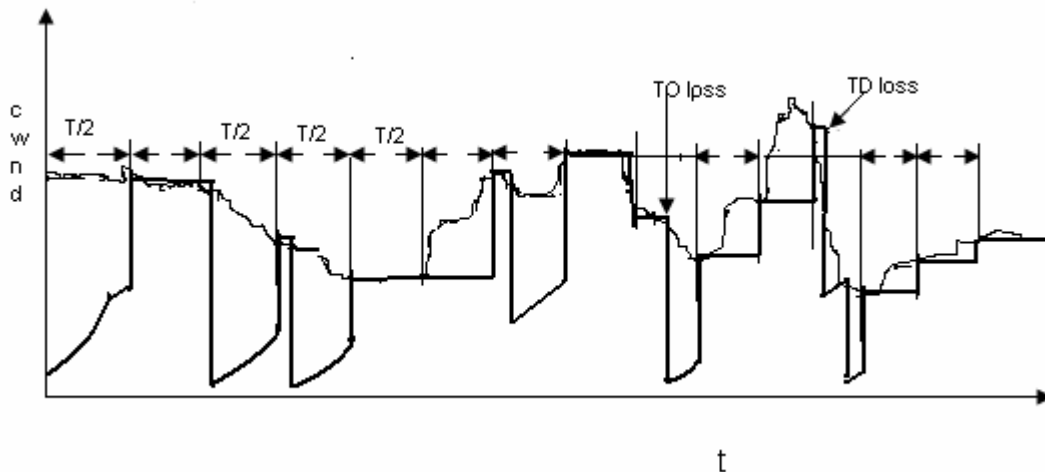


Fig 10: Modified TCP congestion control algorithm

So decreasing the duration of interval allows to sender to get information about network more frequently. We can expect that decreasing the duration of interval allow a TCP connection to use more of its available bandwidth and theoretically it is possible for modified algorithm to perform like idle one for sufficiently small duration of interval. But, is it possible to decrease the observation time (duration of interval) to arbitrarily small value? Answer is NO. Because, if the duration of interval is smaller than a certain range then receiver can not estimate the available bandwidth of network properly, because it will then receive a small number of packet pair's and it has to estimate band width based on these small number of packet pair's information, and that can make wrong estimation of network capacity. So, receivers need to choose an optimal duration of interval that will be enough for receiver to estimate the network approximately and allow the TCP connection to use bandwidth more effectively. The modified algorithm operates as follows:

- Time scale is divided into slots of equal interval.
- Sender sends its packet as packet pair to receiver.

- During each slot receiver estimates available bandwidth (R') for every arrival of packet pair by using the equation $R'=b/t_0$. Here, b is packet size in bytes and t_0 is time gap between two packets of a packet pair to be received at the receiver end.
- For every end of a slot, the receiver estimates the bandwidth R as the minimum R' during this slot and sends this estimated bandwidth (R) of the connection to the sender.
- By the start of next slot, sender adjusts its congestion window size ($cwnd$) to $(R \times RTT)/b$. Here RTT is round trip time.
- Sender sticks to this window size until it finds any congestion in network, and decreases its window size immediately after observing the congestion. Decreasing of window depends on what kind of congestion sender noticed. If TD loss detected then sender decreases its window to half of the current window size and if TO loss detected then sender decrease its window size to one segment.
- After this decrement of window size sender starts its conventional TCP algorithm (TCP Reno) till the end of the current slot and again modify its congestion window size by the start of the next slot.

2.4 Analytical Characteristics of Modified

Algorithm

In Modified algorithm, total transfer time is divided into slots of equal duration T . In each slot sender initially maintains a fixed window size until it finds congestion and then it adapt its window size based on conventional TCP algorithm till the end of the current slot. An i_{th} slot is shown in Figure (11). In this figure $T_2(i)$ denotes the duration

during which sender run conventional TCP algorithm and $T_1(i)$ denotes the duration during which sender tries to stick with the constant window size $W(i)$. Sender calculates $W(i)$ based on the information about network condition it gets from receiver side. $X(i)$ indicates the round number where packet loss occur in $T_1(i)$ time duration and $a(i)$ is the first packet lost in this duration.

Let, $Y(i)$ is the number of packet sent in i_{th} slot and $A(i)$ the duration of slot. As all of the slot's duration are T , so $A(i)=E[A]= T$.

Therefore we can deduce the throughput for this model as

$$B_{Modified} = \frac{E[Y]}{T}. \quad (2.27)$$

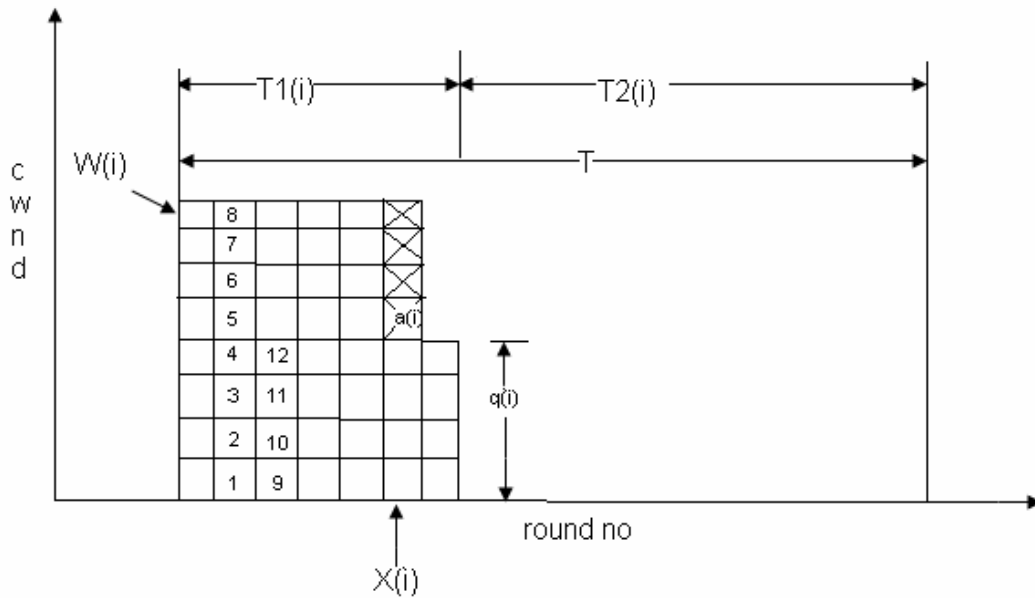


Fig 11: Packet sent during an interval

Now, we divide the total number of packet sent $Y(i)$ in i_{th} slot into $Y_1(i)$ and $Y_2(i)$, where $Y_1(i)$ and $Y_2(i)$ denote number of packet sent in $T_1(i)$ and $T_2(i)$ durations respectively. So,

$Y(i) = Y_1(i) + Y_2(i)$. As $\{Y(i)\}$, $\{Y_1(i)\}$, $\{Y_2(i)\}$ are random processes, so the above equation becomes,

$$E[Y] = E[Y_1] + E[Y_2] \quad (2.28).$$

As total number of packets sent in first round of any slot is chosen based on the available bandwidth of the network, therefore probability of loss for any packet in the first round is much smaller than p which is the probability that a packet is lost in any other round except first round. For simplicity we consider that all packets in first round of each slot reach successfully to receiver side. So, we start the numbering of packet from the second round of each slot instead of first round. After packet $a(i)$, $W(i)-1$ more packets are sent in next round before a loss indication occurs as discussed in section 2.2.2.2. So total number of packet sent in $T_1(i)$ duration (or $X(i)+1$ round) is

$$\begin{aligned} Y_1(i) &= W(i) + a(i) + W(i) - 1 \\ &= a(i) + 2W(i) - 1. \end{aligned}$$

It follows that,

$$E[Y_1] = E[a] + 2 E[W] - 1 \quad (2.29)$$

In section 2.2.2.1, the value of $E[a]$ was derived. From equation (2.5) and (2.29) we get,

$$E[Y_1] = \frac{1-p}{p} + 2E[W] \quad (2.30)$$

From Figure (11) we can observe that,

$$T = T_1(i) + T_2(i) \quad \text{and} \quad T_1(i) = \sum_{j=1}^{X(i)+1} r(i, j)$$

Where $r(i,j)$ denotes round trip time of j th round of i th slot . Here $r(i,j)$ are random variables. So the above equation becomes,

$$E[T_1] = (E[X] + 1) \times RTT$$

From this equation we get the expectation value of X is,

$$E[X] = \frac{E[T_1]}{RTT} - 1 \quad (2.31)$$

Again from Figure (11) we can say that total number of packet sent in $T_1(i)$ duration is,

$$Y_1(i) = X(i)W(i) + q(i).$$

For simplicity let's assume $\{X(i)\}$ and $\{W(i)\}$ are mutually independent sequences of i.i.d random variables.

So,
$$E[Y_1] = E[X]E[W] + E[q] \quad (2.32)$$

As, $q(i)$, the number of packet in the last round in $T_1(i)$ duration is uniformly distributed between 1 and $W(i)$ then,

$$E[q] = \frac{E[W]}{2} \quad (2.33)$$

From equation (2.31),(2.32) and (2.33), we get,

$$E[Y_1] = \left(\frac{E[T_1]}{RTT} - 1 \right) E[W] + \frac{E[W]}{2} \quad (2.34)$$

From equation (2.34) and (2.30) we get the expectation value of T_1 as follows,

$$E[T_1] = RTT \left[\frac{1-p}{pE[W]} + 2.5 \right] \quad (2.35)$$

As, sender run conventional TCP algorithm (TCP Reno) in $T_2(i)$ duration, so its transmission rate B in this duration is same as equation (2.26). So, total number of packet sent in $T_2(i)$ is,

$$Y_2(i) = T_2(i) \times B = (T - T_1(i)) \times B$$

It follows that,

$$E[Y_2] = (T - E[T_1]) \times B \quad (2.36)$$

Average number of packet sent in $E[T_1]$ duration is,

$$E[Y_1] = E[W] \left(\frac{E[T_1]}{RTT} - \frac{1}{2} \right) \quad (2.37)$$

Equation number (2.37) comes from equations (2.31),(2.32) and (2.33).

So, expected value of total number of packet Y sent is,

$$\begin{aligned} E[Y] &= E[Y_1] + E[Y_2] \\ &= E[W] \left(\frac{E[T_1]}{RTT} - \frac{1}{2} \right) + (T - E[T_1])B \end{aligned} \quad (2.38)$$

From equation (2.27) and (2.38) we get the throughput of modified model is,

$$B_{Modified} = \frac{E[W] \left(\frac{E[T_1]}{RTT} - \frac{1}{2} \right) + (T - E[T_1])B}{T} \quad (2.39).$$

Here $E[W]$ is same as equation (2.14).

Chapter 3

Theoretical and Simulation results

In this chapter, Modified TCP algorithm is compared to conventional algorithm (TCP Reno) by Simulation and theoretically, and it will also be shown that how the implementation of TCP friendly rate corresponding to Modified TCP algorithm into ERA can improve the receiving rate of Multicast receivers. Theoretical and simulation comparisons between conventional TCP and Modified TCP are shown in section 3.1 and 3.2 respectively and ERA improvement is shown in section 3.3. In theoretical comparison we consider equations (2.18) and (2.26) for conventional algorithm and equation (2.39) for Modified algorithm. In equation (2.18) ,only TD loss is considered and in equation (2.26) TD loss and TO loss both are considered. Here, it is assumed that receiver has a large buffer size. So sender congestion window is not restricted by receiver's buffer size.

3.1 Theoretical Comparison

In Figure (12), Modified TCP algorithm is compared to Conventional TCP algorithm for different values of 'p'. In this figure equation (2.26) is taken for

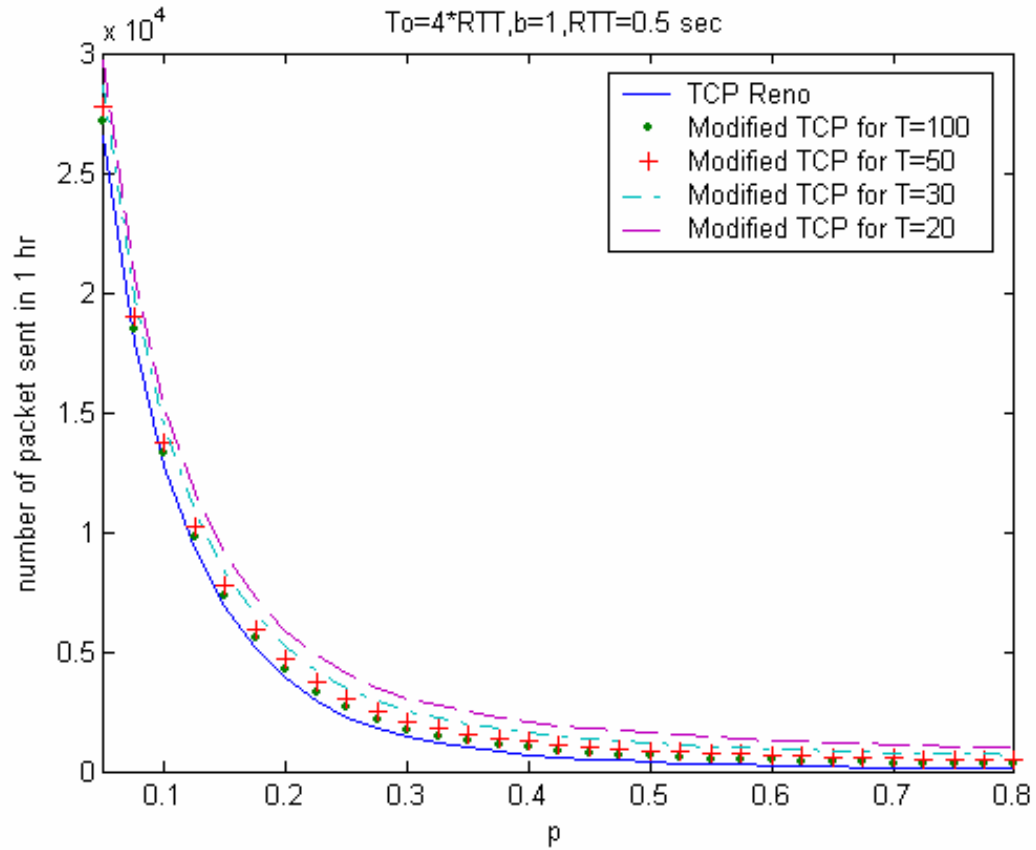


Fig 12: Theoretical comparison between conventional TCP & Modified TCP

Conventional TCP algorithm i.e. TD and TO both loss are considered. The x-axis represents the probability of loss indications, p , while y-axis represents number of packet sent in one hour durations. From this figure, it can be observed that Modified algorithm gives better performance than conventional one for each value of p , and its performance improves for smaller values of slot duration (T). In Figure (13) it is explicitly shown that how performance of Modified algorithm improves for different values of T .

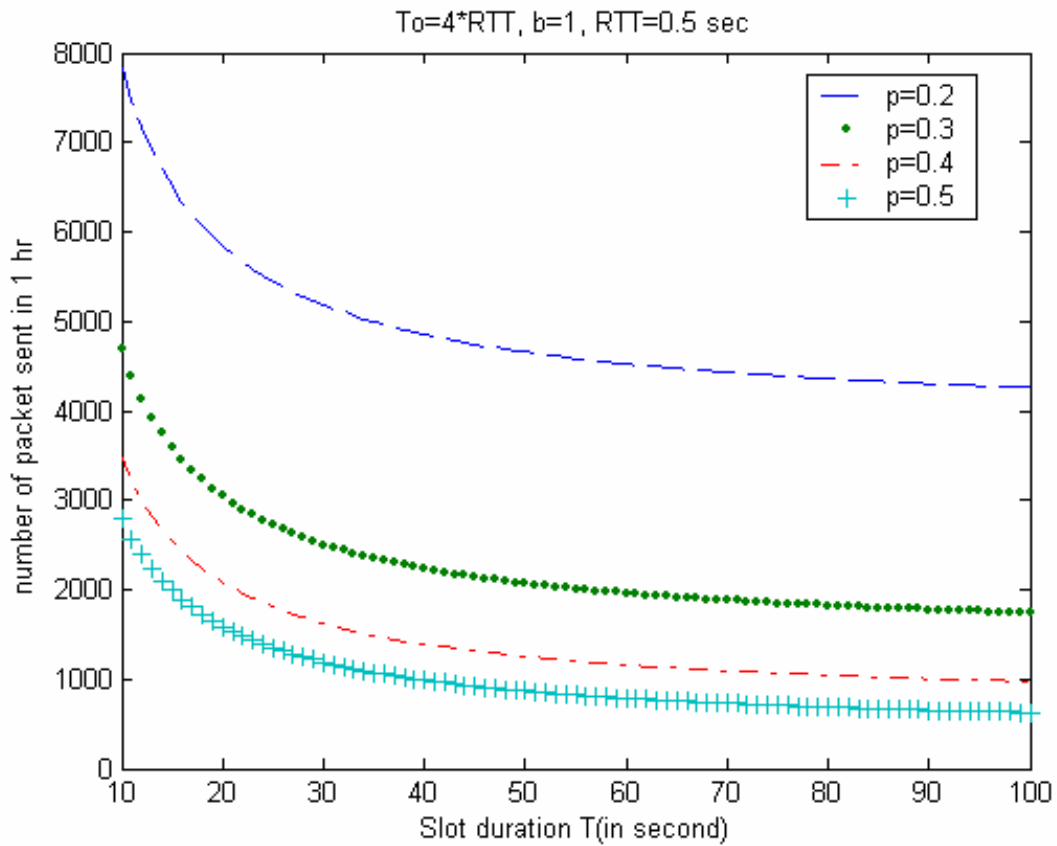


Fig 13: Theoretical results of Modified TCP for TD & TO loss

In Figure (13), x-axis represents the slot duration T in sec, and y-axis represents number of packet sent in one hour in Modified TCP. In this figure four plots are taken based on four different 'p'. It is seen from the above figure that with decreasing T number of packet sent is increasing. Figure (14) shows similar plot. But, here equation (2.18) is considered for conventional TCP algorithm which is used to estimate bandwidth of Modified TCP.

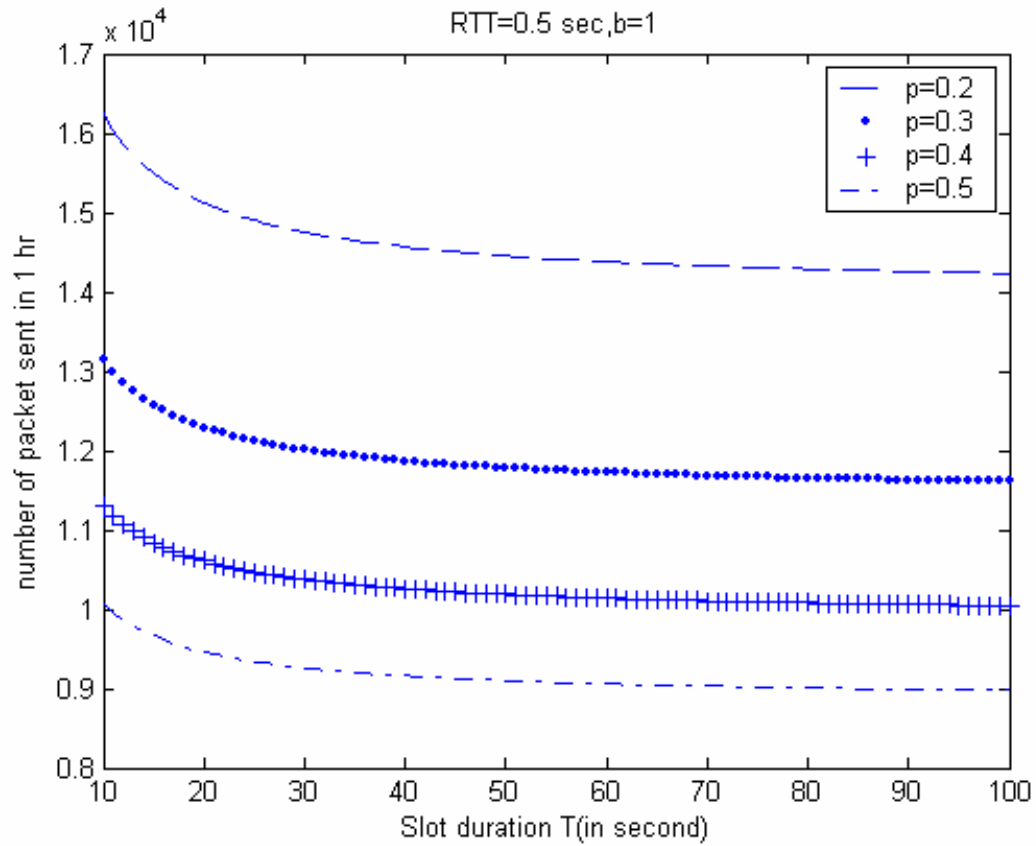


Fig 14: Theoretical results of Modified TCP for TD loss only

From Figure (13) and (14), it is seen that total number of packet sent in one hour in Figure (14) is higher than in Figure (13). This is because equation (2.18) overestimates available bandwidth for greater than 5% loss i.e. $p=0.05$. This was described in last chapter (sec 2.2.2)

3.2 Simulation Results

In this section, Modified algorithm's performance is compared with Conventional algorithm's performance by simulation, and then, simulation results are compared with the theoretical results. For the purpose of simulation, some assumptions are taken. These assumptions are:

- Congestion is considered only for data packets, no congestion for ACK packets.
- Simulation is approximated in terms of rounds.
- It is also assume that packet losses within a round are correlated, i.e. if any packet is lost in any round then remaining packets in that round after first packet lost are also lost.
- Packet loss in one round is independent of loss in other rounds.
- Whole simulation is carried out by taking two cases separately. In first case, only TD packet loss is considered, i.e. for each packet loss, sender reduces its window size by factor of two. In second case, both TD and TO losses are considered.

In Figure (15), (16) and (17), simulation results are shown for the first case, i.e. only TD losses are considered. In Figure (15), comparison between Modified TCP algorithm and Conventional algorithm is shown in terms of throughput in one hour. It can be seen from above figure that Modified algorithm has higher throughput than conventional one and the throughput of Modified algorithm is increasing with decreasing the duration T. In Figure (16), another comparison is shown between Modified TCP and Conventional TCP in terms of Good put in one hour duration transfer. Here Good- put signifies how many different numbers (non- duplicate) of packets are sent.

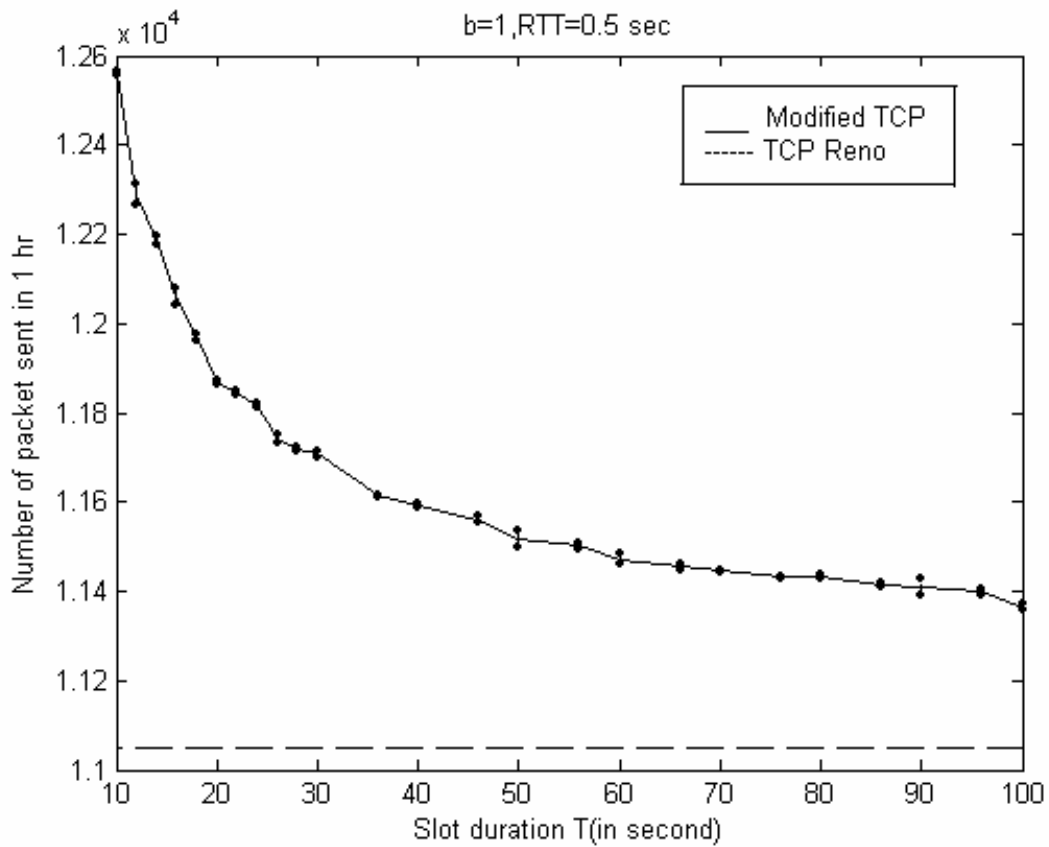


Fig 15: Throughput comparison for TD loss indication

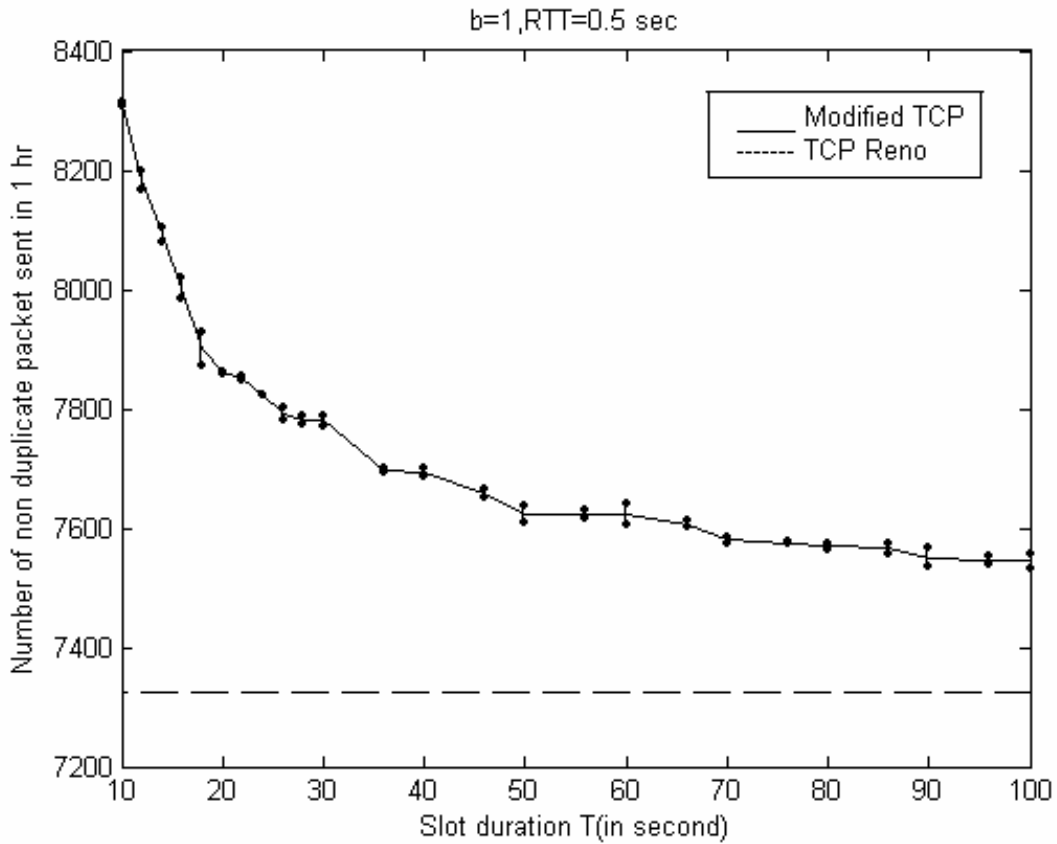


Fig 16: Goodput comparison for TD loss indication

In Figure 17, simulation results for Modified TCP algorithm is compared with its theoretical results obtained by using equation (2.18). In this comparison round trip time RTT is taken 0.5 and the value of ‘p’ is approximated by the ratio of total number of packet losses to the total number of packet sent in one hour data transfer. .Number of packet losses and number of packet sent in one hour is taken from simulation results. In this simulation packet loss rate ‘p’ is taken near about 0.3. From Figure (17) it can be observed that theoretically computed results are nearly similar to the simulation results.

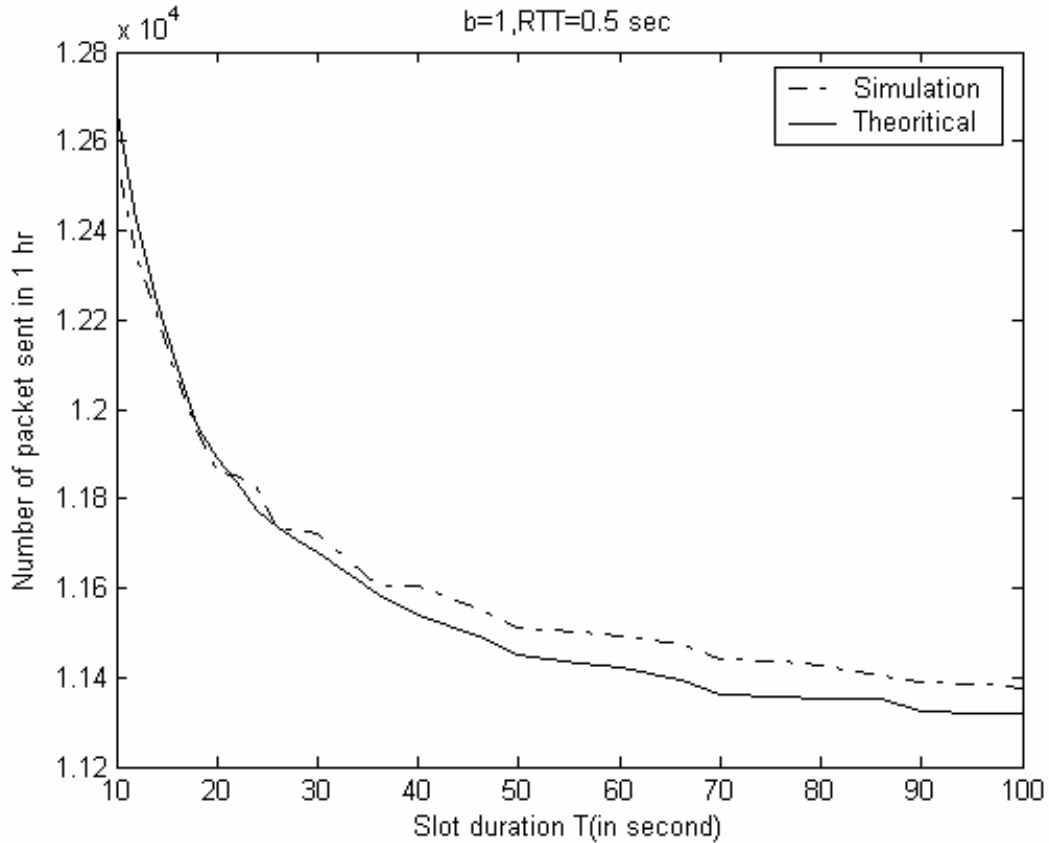


Fig 17: Simulation & Theoretical comparison for TD loss indication

In Figure (18), (19) and (20), simulation results are shown for the case where TD and TO both losses are taken into account. In Figure (18), Modified TCP's throughput is compared with TCP Reno's throughput. Here, sender time out T_0 is taken four times of RTT. In Figure (19), TCP Reno and Modified TCP are compared based on their good put in one hour data transfer. From Figure (18) & (19) it can be observed that Modified TCP works better than conventional TCP algorithm (TCP Reno), and performance of Modified TCP is improving with decreasing of T(slot duration).

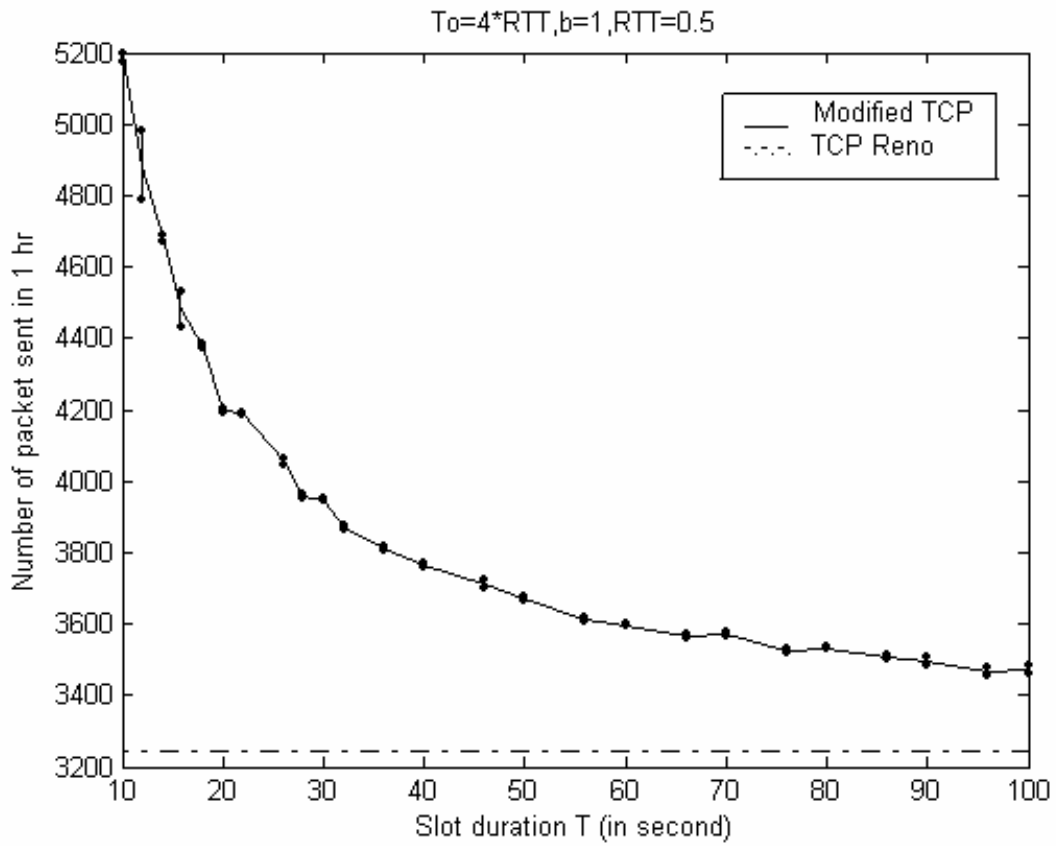


Fig 18: Throughput comparison for TD & TO loss indication

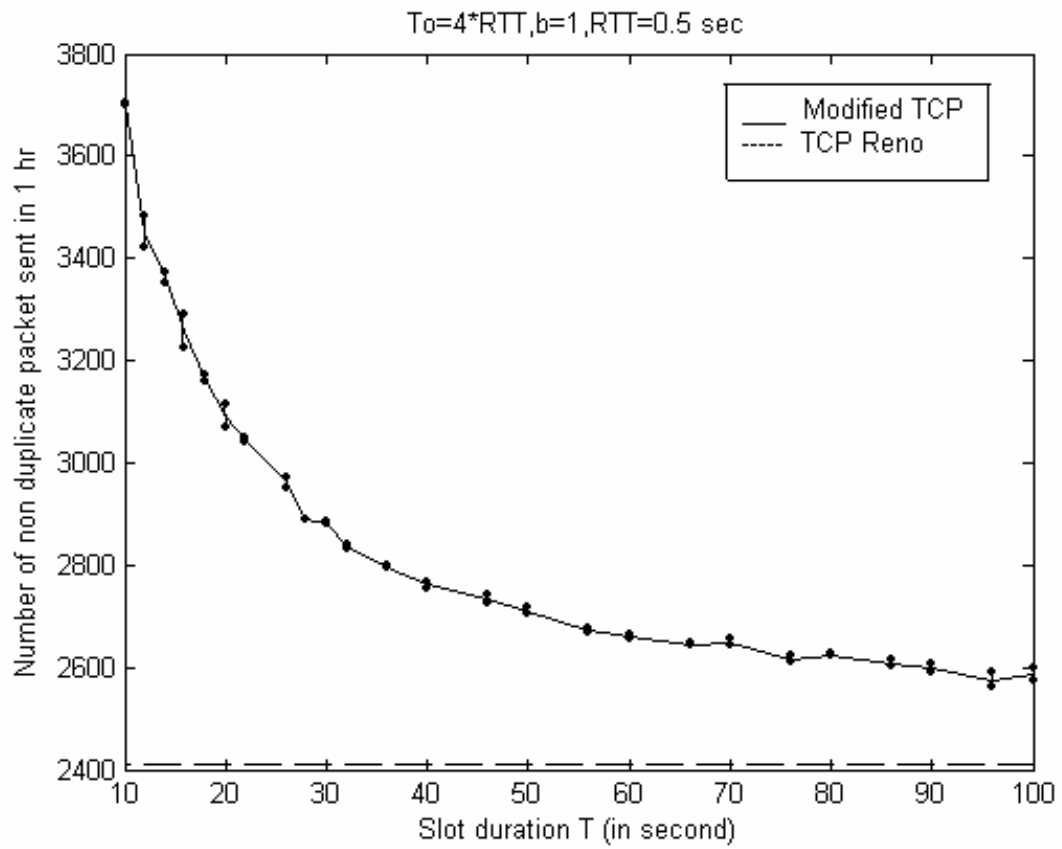


Fig 19: Goodput comparison for TD & TO loss indication

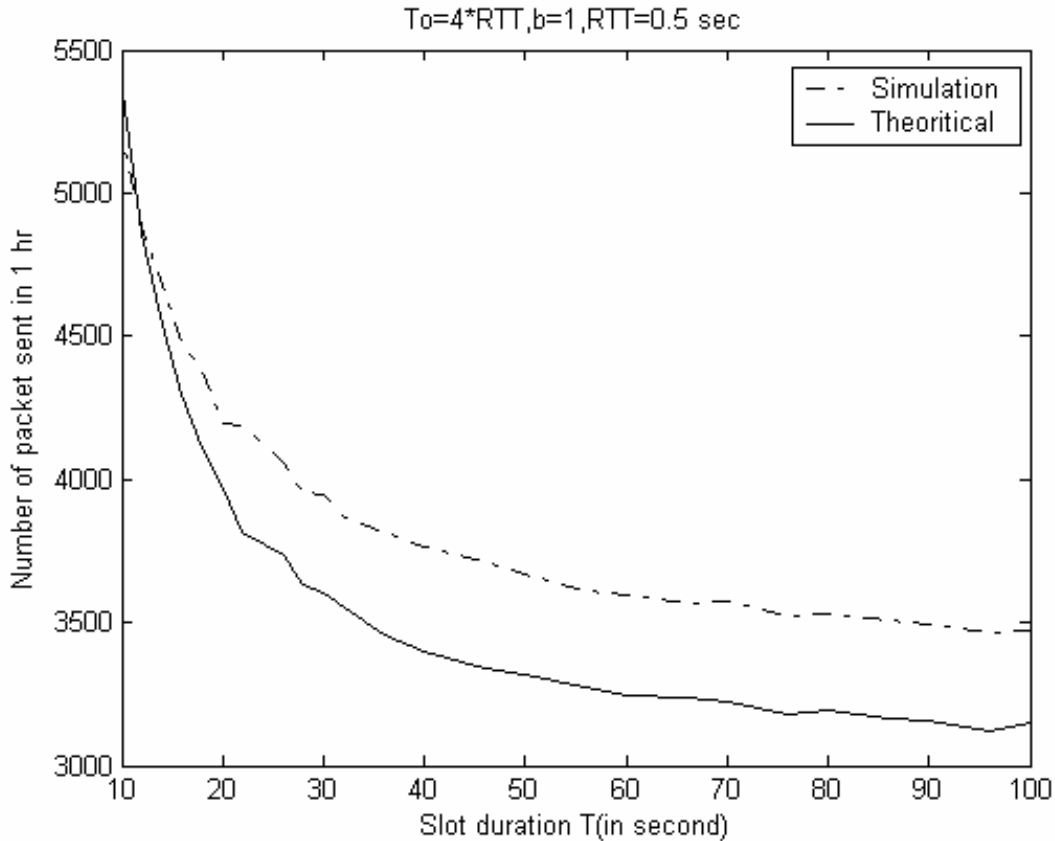


Fig 20: Theoretical & Simulation comparison for TD & TO loss indication

In Figure (20), simulation results are compared with theoretical results. From this comparison it can be observed that theoretical results approximately support simulation results.

Till now, packet loss is considered only when congestion occurs in network. This is true for wired network, but this can not be considered in the scenario where packets are lost due to noise. Wireless network is an example of this scenario. If packet loss due to noise is included then performance of TCP connection will go down, because from sender perspective this loss could be due to the congestion occurrence in network and sender reduces its rate unnecessarily. This is shown in Figure (21). In this simulation TD and TO

both losses are considered. From this figure it is observed that throughput of TCP is decreasing with increasing the noise loss probability p' .

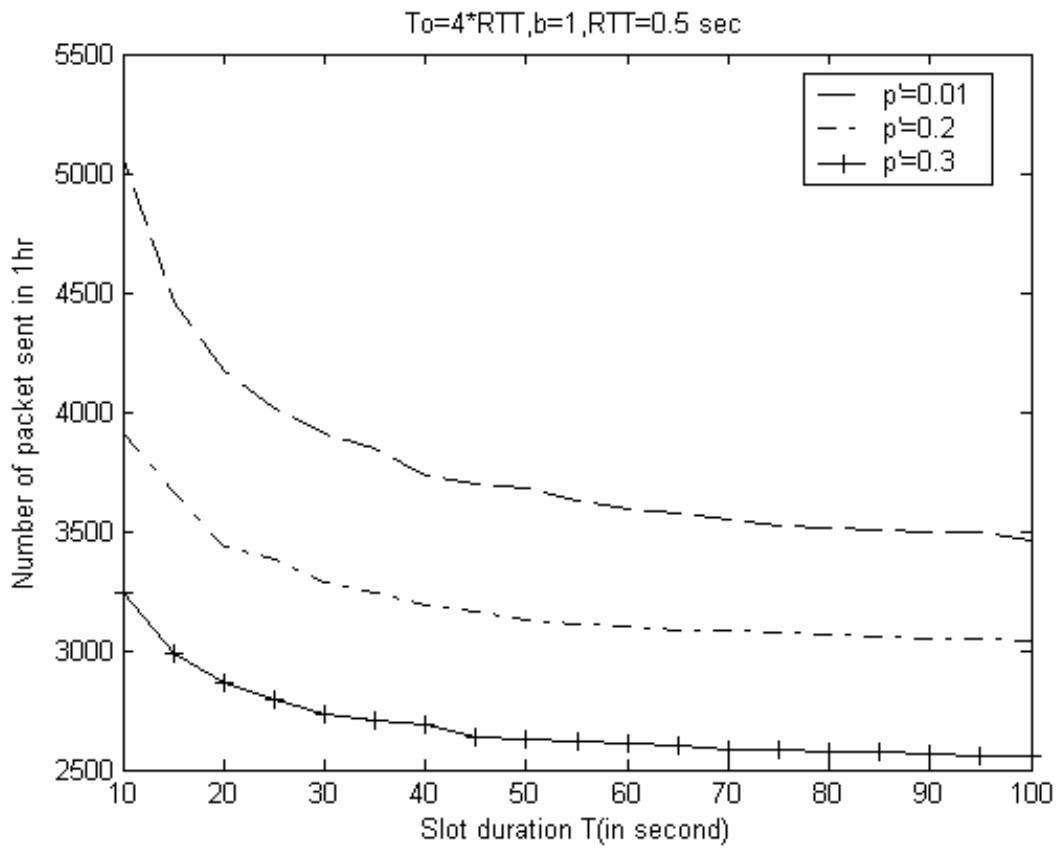


Fig 21: Performance of Mod. TCP for packet loss due to noise

3.3 Explicit Rate Adjustment (ERA) Improvement

In this section, it is shown that how ERA algorithm can improve receiving rate of Multicast receivers by using equation (2.39) which is TCP friendly rate derived from Modified TCP algorithm. According to ERA algorithm sender distributes its signal into number of layers. Joining more layers signifies more receiving rate i.e. if any receiver joins higher number of layer then its receiving rate also be higher.

Here, for purpose of comparison a complete session having 20 Rate adjustments Interval (RAI) is considered. Sender transmits its signal at rate of 12800 bytes /second and it distributes its signal into 21 numbers of layers. Lower layer carrying obligatory data to decode the signal and it carries data at 2000 bytes/second. It is assumed that remaining data are uniformly distributed in other 20 layers. Figure (22) shows the improvement of receiving rate. X-axis indicates RAI number of session and Y-axis represents the layer number. Each point which is indicated by either 'o' or '+' in this figure represents layer number for corresponding RAI. The points are joined by line segments only for better visual representation of the data. Here, 'o' and '+' are used to indicate points for ERA algorithm using equations (2.39) and (2.26) respectively. This figure shows that for each RAI, improved ERA chooses higher layer than previous one.

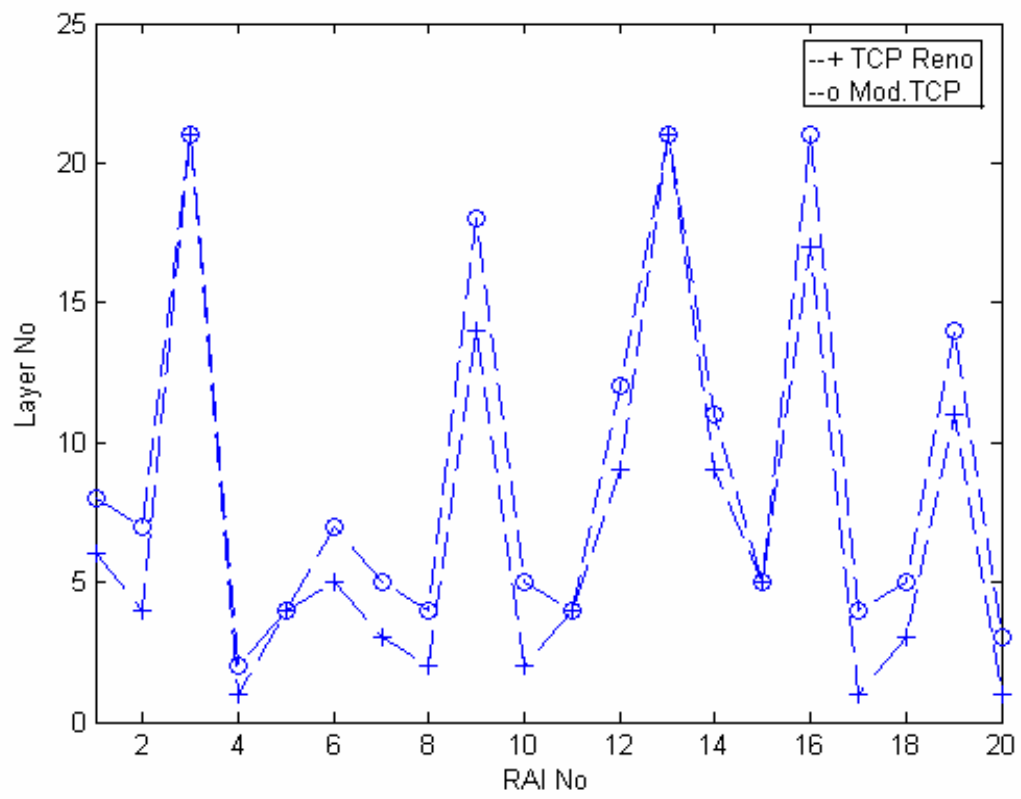


Fig 22: Comparison between ERA and improved ERA

Chapter 4

Conclusion

This thesis presented a modification in TCP Reno and its use in Explicit rate adjustment (ERA) algorithm. In TCP Reno, sender adjusts its cwnd size by blindly monitoring the network. Whereas, in modified TCP sender gets information about network capacity from receiver side. Receiver estimates network bandwidth by packet pair (PP) technique. ERA algorithm is modified by using modified TCP algorithm throughput as TCP friendly rate for estimating target rate of receivers.

The throughput of modified TCP was computed analytically and then it was compared with the analytically estimated throughput of TCP Reno. The performance comparison was made in terms of average number of packet sent in one hour data transfer. The performance comparison was also shown by carrying out simulation. From Figure (12)-(20), it is observed that performance of modified TCP is better than TCP Reno.

The performance comparison was also shown between ERA using TCP Reno estimate and ERA with modified TCP estimate algorithm. From Figure (21), it is seen that ERA with modified TCP Reno performs better than before.

4.1 Future Aspects

In wireless connection, packet losses occur not only for congestion in network, but due to damaging of packet by noise too. In TCP algorithm, sender always takes a packet loss as occurrence of congestion in network, so it slow down its rate which is unnecessary for

packet losses due to packet distortion by noise in wireless channel. So, conventional TCP's performance can not be satisfactory in wireless connection, whereas Modified TCP can do better work than conventional TCP, because in Modified TCP sender gets knowledge of network capacity. This is shown in Figure (21), but for this simulation it is assumed that feedback packet from receiver which carries information of network capacity always reach to sender successfully. This assumption can be taken for wired connection but it can not be taken in wireless scenario, because packet loss probability is higher in wireless connection, so these feedback packets can also be lost. So, sender will be unable to get information of network capacity at end of each slot.

It is believed that if receiver sends more than one feedback instead of one at the end of each slot then chances of receiving at least one feedback packet by sender will increase which should increase the performance of Modified TCP in wireless connection. This simulation is not done in this thesis and can be done as future work.

References

- [1] DEERING, S.E. Multicast Routing in a Datagram Internetwork. PhD thesis Stanford University, Dec 1991.
- [2] ERIKSSON, H.Mbone: The multicast backbone Communications of the ACM 37, 8(1994), 54-60.
- [3] BOLOT, J.C., TURLETTI, T., AND WAKEMANI. Scalable feedback control for multicast video distribution in the Internet. In Proceedings of SIGCOMM'94 (University College London, London, U.K., Sept.1994), ACM.
- [4] GILGE, M., AND GUSELLA, R. Motion video coding for packet switching networks an integrated approach. In proceedings of the SPIE conference on visual communications and Image Processing.
- [5] KANAKIA,H.,MISHRA, p.p., AND REIBMAN,A. An adaptive congestion control scheme for real time packet video transport. In proceedings of SIGCOMM'93(San Francisco, CA, sept 1993). ACM, pp20-31.
- [6] SHACHAM, N. Multipoint communication by hierarchically encoded data. In Proceedings IEEE.
- [7] TAUBMAN,D., AND ZAKHOR, A. Multirate 3-D subband coding of video. IEEE Transaction Image processing 3, 5(sept 1994), 572-588.
- [8] DEERING,S.. Internet multicast Routing: state of the art and open research issues, Oct 1993. Multimedia Integrated conferencing for Europe (MICE) seminar at the Swedish Institute of computer science, Stockholm.

- [9] CHADDHA,N., AND GUPTA, A. frame work for live multicast of video streams over the Internet. In proceedings of the IEEE International Conference on Image processing (Lausanne, Switzerland, sept, 1996).
- [10] DELGROSSI, L., HALSTRICK, C., HEHMANN, D., HERRTWICH, R.G., KRONE, o., SANDVOSS, J., AND VOGT, C. Media scaling for audiovisual communication with the Heidelberg transport system. In proceedings of ACM Multimedia 93, ACM, pp. 99-104.
- [11] HOFFMAN, D., AND SPEER, M. Hierarchical video distribution over Internet style networks. In proceeding of the IEEE International conference on Image processing (Lausanne, Switzerland, sept, 1996).
- [12] MCCANNE, S., AND VETTERLI, M. Joint source/channel coding for multicast packet video. In proceedings of the IEEE International conference on Image Processing (Washington, DC, oct, 1995).
- [13] SPEER, M.F., AND MCCANNE, S. RTP Usage with Layered Multimedia streams. Internet Engineering Task force, Audio Video Transport working group, Mar, 1996.
- [14] Explicit rate Adjustment: an Efficient Congestion Control Protocol for Layered Multicast. S.Puangpronpitag, R.Boyle and K.Djemame.
- [15] J.Mahdavi and S.Floyd, "TCP-friendly Unicast rate based Flow Control", Technical note sent to the end2end-interest mailing list, January 1997, <http://www.psc.edu/networking/papers/tcpfriendly.html>.
- [16] J.D.Padhye, V.Firoiu, D.F. Towsley, and J.F.Kurose, "Modeling TCP throughput: A simple Model and its empirical validation", In processing of ACM SIGCOMM, pp.303-314, Vancouver, Canada, September 1998.

- [17] V. Paxson, “End-to-end Internet Packet Dynamics”, Computer communication vol.27,iss.4,pp,139-152, October 1997.
- [18] V.Paxson, “Measurement, and Analysis of End-to-end Internet Dynamics”, PhD thesis Lawrence Berkeley National Laboratory, Universities of California, Berkeley, California, USA, 1997.
- [19] M.Luby, V.K. Goyal, and S, Skaria, “wave and equation based rate control: A Massively Scalable Receiver Driven Congestion Control Protocol”, Computer Communication, vol.32, iss.4, pp.191-214, October 2002.
- [20] S.McCanne,V.Jacobson, and M.Vetterli,“Receiver driven Layered Multicast “, In proceedings of ACM SIGCOMM, vol.26,pp.117-130, New York, USA,August 1996.
- [21] Tanenbaum, A.S. Computer Networks. Pearson Education, Asia-2003.
- [22] W.Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithm. RFC 2001, Jan 1997.
- [23] L.Vicisano, L.Rizzo, and J.Crowcroft, “TCP like congestion control for Layered Multicast data Transfer”, In Proceedings of IEEE INFOCOM, pp.996-1003, San Francisco, USA, April 1998.
- [24] A.Legout and E.W. Biersack,” PLM: Fast Convergence for Cumulative Layered Multicast Transmission”, In Proceedings of ACM SIGMETRICS, pp. 13-32, Santa Clara, California, USA, June 2000.
- [25] D.Sisalem and A. Wolisz, “MLDA: A TCP friendly Congestion Control scheme”, In Proceedings of International Workshop on Quality of service, pp.65-74, Pittsburgh, PA, USA, June 2000.