TCP based Receiver Assistant Congestion Control

Hardik K. Molia
*Master of Computer Engineering,
Department of Computer Engineering
Atmiya Institute of Technology & Science,
Rajkot- 360005,
Gujarat, India*

Prof. Rashmi Agrawal
*Assistant Professor,
Department of Computer Engineering
Atmiya Institute of Technology & Science,
Rajkot- 360005,
Gujarat, India*

Abstract— **Transmission Control Protocol (TCP) is a transport layer protocol which provides process to process communication. TCP is a connection oriented and reliable protocol. TCP provides stream communication with the help of ordered delivery concept. TCP breaks the data coming from the higher layers into a set of segments, it assigns a sequence number and sends to the network layer for further processing. Network layer adds IP header to individual segment and let them propagate through the network to reach the receiver via the lower layers. Congestion is the situation when the load of the network (no. of packets to handle) is larger than its capacity. Congestion leads to discard of some of the packets which will degrade the performance. This paper shows a novel scheme to handle congestion in which receiver takes part to inform the sender.**

Keywords— ***TCP, Congestion, Slow Start, Congestion Avoidance, Fast Retransmission, Fast Recovery.***

I. Introduction

Every computer runs many processes at a time and when we communicate from one computer to another computer, inherently two processes running on each of the communicating computers are exchanging data. While Data link layer is responsible for node-to-node delivery, network layer is responsible for host-to-host delivery; transport layer is responsible for end-to-end or process-to-process delivery. Transport layer identifies each point of communication via a socket address. Socket address is a combination of IP Address to identify the node and Port Address to identify the process running on that host. One of the most widely used transport layer protocol is Transmission control protocol (TCP).

TCP provides accurate delivery rather than timely delivery. TCP may introduce long delays to handle out-of-order packets or retransmissions of lost packets. While IP handles actual delivery of the data, TCP keeps track of the individual units of data transmission, called segments that a message is divided into for efficient routing through the network. For example, when HTML file is requested from the web server, the server TCP breaks the HTML data into set of segments, inserts TCP headers and forwards them individually to server IP. IP adds IP header to each of these segments and let them propagate through the network to reach to the client computer. Client TCP reassembles individual segments and ensure that they are ordered and error free.

TCP provides following services.

1. Process to Process Delivery:- Communication among two processes running on two different nodes.TCP identifies each end of the communication with a socket address which is a combination of IP address and port address.

2. Stream Communication:-TCP receives data from the higher layers and split it into set of segments. TCP also defines relationship among segments by including a sequence number which is byte oriented. TCP running on the receiver guarantees that the data will be provided back to the actual receiver in the same order in which it was sent.

3. Full Duplex: - TCP supports full duplex communication. This is bidirectional communication.

4. Flow Control: - Receiver can sends the rate at which it can receive data to the sender as part of ACK packet (inside header field of the ACK TCP segment). So sender can slow down the sending rate and receiver will not suffer from overwhelming due to full of buffers.

5. Error Control: - TCP identifies errors using error detection and correction mechanisms to identify corrupted packets. Retransmission is performed for lost packets.

6. Congestion Control: - TCP analyses the network by analysing pattern of packet loss and packet delay and take necessary actions like slow down the sending rate, freezing until congestion is removed.

II. Congestion Control

Congestion because intermediate devices like routers and switches have limited sized buffers to store the packets before and after processing. When a packet arrives at the incoming interface, it undergoes three steps before departing.

1. The packet is put at the end of the input queue.

2. Router program removes a packet from the queue and finds its route.

3. The packet is sent to appropriate output queue and waits its turn to be sent.

There are two issues. First, if the rate of packet arrival is higher than the packet processing rate, the input queues become longer and longer. Second, if the packet departure rate is less than the packet processing rate, the output queues become longer and longer. At this point, because of the packet overflow in any of the buffer queues, routers may discard some packets. If senders still continue to send more and more packets, the situation may become worst. Transport layer and network layer should work together for congestion control. Network layer witnesses the congestion while transport layer causes congestion. Congestion control can be of two types.

1. Open-Loop, Congestion Avoidance, Proactive schemes based on retransmission, window, acknowledgements, discard, admission policies.

2. Closed-Loop, Congestion Detection & Recovery, Reactive schemes based on various techniques like back pressure, choke packet, implicit signalling, and explicit signalling.

A. *Congestion Window*

TCP uses a sliding window to keep track of number of bytes sent and acknowledged, sent but not acknowledged yet, pending to be sent. TCP varies the size of sliding window as per the current situation in the network as well as of the receiver. The actual size of the sender sliding window wnd is maximum number of outstanding bytes that can be sent without expecting any acknowledgement. on occurrences of acknowledgements, TCP increases size of the sliding window. A round is the completion of transmission of all the bytes presently loaded into the window.

wnd = MIN(rwnd, cwnd)

Receiver sends the maximum rate at which it can receive by specify the window size parameter in TCP header of the acknowledgement or piggybacked data. This is known as rwnd which is a part of flow control. Sender uses various signals like packet delay, packet loss, and pattern of acknowledgements to predicate the congestion of the network. Based on such prediction, sender maintains a value of congestion window, cwnd which is a part of congestion control.

B. *Congestion Policy*

TCP's congestion control is based on three phases: slow start (exponential increase), congestion avoidance (additive increase), and congestion detection (multiplicative decrease). In the slow-start phase, the sender starts with a slow rate of transmission, but increases the rate exponentially until it reaches a threshold value. At the threshold value, congestion avoidance phase starts where TCP increases the sending rate linearly. If congestion is detected at any point, the sending rate is reduced and either slow start or congestion avoidance phase is started based on how the congestion was detected.

C. *Slow Start-Exponential Increase*

In slow start, size of the congestion window is increased exponentially per round completion. At the time of connection establishment, congestion window cwnd is set to very few segments, Mostly 1 to 4. (1 - Maximum segment size). The congestion control scheme is byte oriented but for simplicity we are considering segments as unit of different windows.

Receiver sends an acknowledgement for successfully recived packets. Sender will increment the size of congestion window by 1 for every successful acknowledgement. So eventually cwnd will be doubled for every round – per round trip time.

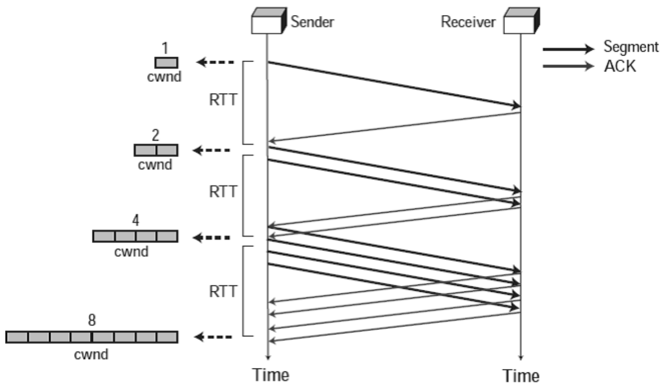

Fig 1. Slow Start

A threshold value – ssthresh has been decided in starting which is mostly equal to half of the maximum window size supported by the system. Sender continues with the slow start until the size of congestion window cwnd reaches the ssthresh. At this point, it enters into congestion avoidance. In the slow-start algorithm, the size of the congestion window increases exponentially after every round 1,2,4,8,…, after every ACK, 1,2,3,4,5…. until it reaches a threshold.

The main reason of slow start is not to flood the network immediately with lots of packets in beginning without caring about the current situation.

In slow start, size of the congestion window (cwnd) starts with one maximum segment size (MSS). The MSS is determined during connection establishment by using an option of the same name. The size of the window increases one MSS each time an acknowledgment is received. As the name implies, the window starts slowly, but grows exponentially. Slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named ssthresh (slow-start threshold). When the size of window in bytes reaches ssthresh, congestion avoidance phase gets started. In most implementations the value of ssthresh is 65,535 bytes.

D. *Congestion Avoidance-Additive Increase*

Congestion avoidance is a proactive effort towards the congestion control. To avoid congestion, we must reduce the rate before congestion happens. Slow start's exponential growth is replaced with congestion avoidance's linear growth known as additive increase. In this algorithm, cwnd is not incremented with every acknowledgement, but it is incremented with every round- when whole window of segments is acknowledged. In the congestion avoidance algorithm, the size of the congestion window increases additively after every round 1,2,3,4,5…until congestion is detected.

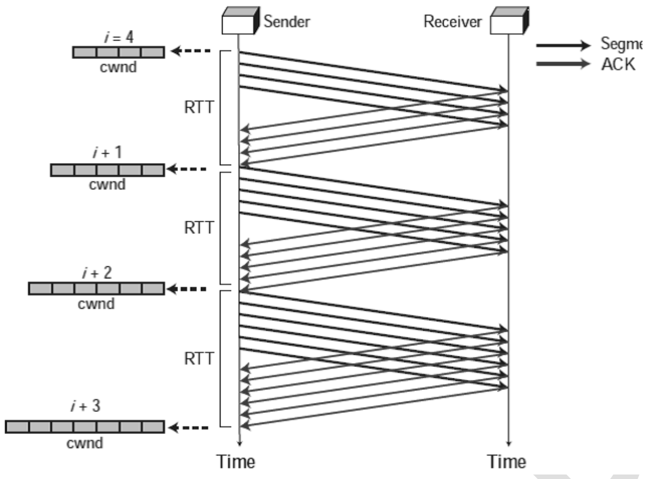

Fig 2. Congestion Avoidance

E. *Congestion Detection-Multiplicative Increase*

If congestion occurs, the congestion window size must be decreased. Retransmission can occur in one of two cases: when a timer times out or when three ACKs are received. In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease. Most TCP implementations have two reactions:

I. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments.

In this case TCP reacts strongly:
a. It sets the value of the threshold to one-half of the current window size.
b. It sets cwnd to the size of one segment.
c. It starts the slow-start phase again.

If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction:

a. It sets the value of the threshold to one-half of the current window size.
b. It sets cwnd to the value of the threshold (some implementations add three segment sizes to the threshold).
c. It starts the congestion avoidance phase.

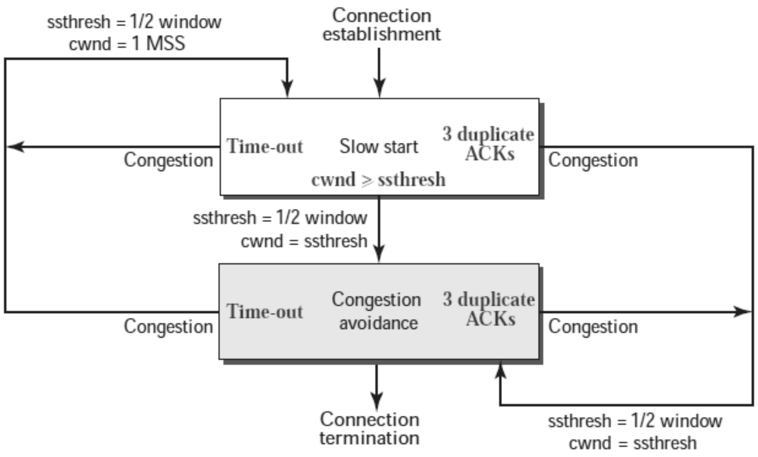

Fig 3. Congestion Detection and Recovery

F. *Congestion Example*

Based on the AIMD - Additive Increase, Multiplicative Decrease concept, following example shows how sender can detect congestion. SS-slow start, AI-additive increase, MD-multiplicative decrease.

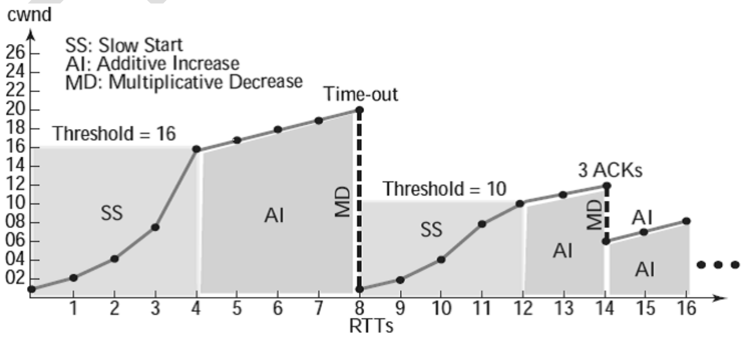

Fig 4. Congestion Example

We are assuming that the advertised window size rwnd will be always much larger than calculated cwnd. Cwnd<<<rwnd. Initially slow start phase begins with cwnd=1 and ssthresh=16.

Slow star phase increments cwnd by 1 with every acknowledgement and so subsequently doubles cwnd with every round up to ssthresh. When cwnd reaches 16, congestion avoidance phase is started which will still increase cwnd by 1 but with every round acknowledgement. When cwnd = 20, re transmission time out occurs which will subsequently, sets ssthresh to cwnd/2 which is 10, cwnd=1 and will start from the slow start.

When cwnd is 12, a 3 duplicate acknowledgement comes which will set ssthresh to cwnd/2 which is 6 and cwnd=cwnd/2 which is 6 as it is fast recovery scheme.

III. Congestion Control Variants

Various TCP congestion control schemes have been proposed for specific kind of network to enhance capability of congestion control.

1. Proactive congestion control schemes where the main goal is to avoid congestion by analyzing packet delay patterns. TCP Vegas is a proactive protocol.

2. Reactive congestion control schemes where the main goal is to detect and recover congestion by analyzing packet loss patterns. TCP Reno, TCP NewReno are reactive protocols.

TCP protocol was initially designed for wireless networks where the probability of congestion loss is much higher than of channel loss due to interference issues. Today wireless networks are becoming more and more popular. Wireless communications are vulnerable to channel loss issue. The standard TCP has no inherent mechanism to identify whether the packet loss is because of the congestion loss or channel loss. By default TCP considers every loss as a congestion loss. TCP's misinterpretation between congestion loss and channel loss will make sender to slow down the rate even when it is not required.

Some advance TCP have some special functionality to work in wireless environment. One such proposed TCP sketch is explained in next section.

IV. Receiver Assisted TCP Scheme

TCP uses a feedback signal to detect / predicate congestion in the network. Feedback signals can be following.

Signal	Detection	Example
Packet Loss	Late	TCP NewReno
Packet Delay	Early	TCP Vegas
Router Indication	Early	TCP Feedback

In standard TCP approach, receiver advertises the windows size (rwnd) as a part of the TCP header of the acknowledgement or piggybacked data in window size field. Receiver's role is limited to the flow control only. The burden of congestion control is on the sender only. In case of some special TCP variants where routers can send feedback regarding their load related issues to the sender.

This paper shows an algorithm to let receiver participate in the congestion control too. The primary feedback signal which we are using is packet delay. A receiver assisted TCP is a combination of standard Sender side congestion control as well as receiver side feedback. Sender continuous with the standard TCP control until it finds a feedback from the receiver.

To provide feedback, algorithm uses the 4 reserved bits of TCP header.

Reserve bits	Error
0000	Everything is OK
0001	packet loss
0010	packet delay

A. *Receiver Side Algorithm*

```
1.  Counter=0
    Packet_delay=0
    Packet_loss=0

2.  Init_Duration = Average time taken a packet by taking
    average of first three packets to reach the destination
    since the connection was established.

3.  For every new successfully received packet n,

    Code=0000

    Packet_n_duration = now – Packet_n-1_duration

    If Packet_n_duration >>>  Init_duration then

            Packet_delay=packet_delay+1
    End if

    If packet_delay = 3 then
            Code=0001
            Exit
    End if

    If on arrival of n^th packet, k previous packets are
    pending to be received then

    If k >=rwnd then
            Code=0010
            Exit
    End if

    If for packet n, code=0000 then use Packet_n_duration
    to find new average. Update Init_Duration

    End for
4.  Send Acknowledgement with code in place of 4
    reserved bits.
```

B. *Sender Side Algorithm*

```
1.  Retrieve the code from the acknowledgement packet.

2.  If code = 0000 then
            Continue with the normal congestion
          control.
      Exit
      End if
```

```
3. If code=0001 then
      //packet delay is detected.

      Go to congestion detection phase with fast
      retransmission technique.
   End if

4. If code=0010 then
   //packet loss is detected.

   Freeze the sending process until an acknowledgement
   with code 0000 code comes. And then continue.

   Freezing time out may be set if the last sent packet
   had sent 0010 code.

   End if
```

VI. Future Expansion

Receiver assistant congestion control scheme can be implemented with any of the TCP variants which supports AIMD concept. As we have 4 bits for feedback, we can also describe various levels delays like moderate delay, high delay and low delay. The scheme is based on packet loss as well as packet delay analysis. Better performance can be achieved if we use it along with router feedback concept.

References

[1] M. Allman, V. Paxson and W. R.Stevens, "TCP congestion control", in IETF RFC, 2581, 1999.
[2] V. Jacobson, "Modified TCP congestion avoidance algorithm. nd2endinterest mailing list", in Tech. Report in IEICE Transactions on Communications, 2007, E90-B(3):516-52, 1990.
[3] T. Bonald, "Comparison of TCP Reno and TCP Vegas: efficiency and fairness", in Performance Evaluation, Vol. 36-37, pp. 307-332, 1999.
[4] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP", in ACM Computer Communication Review, Vol 26, pp. 5-12, 1996.
[5] V. Jacobson, "Congestion avoidance and control", in SIGCOMM Symposium on Communications Architectures and Protocols, pp. 314– 329, 1999.
[6] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP", in Proceedings of SIGCOMM Symposium, pp. 270-280, 1996.
[7] M. Mathis, S. Floyd and A. Romanow, "TCP selective acknowledgment options", IETF RFC, 2018, 1996.
[8] S. Floyd, T. Henderson, and A. Gurtov, "The new Reno modification to TCP's fast recovery algorithm, IETF RFC, 3782, 2004.