TCP Veno

<< A TCP Congestion Algorithm >>

CS415 Wireless Networks End-Sem Seminar

Topic No: - 130

Awanit Ranjan

Roll - 181CO161

National Institute of Technology Karnataka, Surathkal, India

awanitranjan.181me214@nitk.edu.in

Overview

- What is TCP Congestion Algorithm
- Terminologies used in TCP Congestion Control
- TCP Reno
- TCP Vegas
- **TCP Veno** Introduction
- **TCP Veno** Algorithm discussion
- References

What is TCP Congestion Algorithm ?

- Network congestion occurs when packets sent from the source (example - server) exceed what the destination can handle.
- Due to congestion, buffers get filled as the packets are temporarily stored in buffers and this cause several issues like packet loss, retransmissions, reduced data throughput etc.
- TCP is a reliable connection-oriented protocol and it uses a congestion window and a congestion policy that avoid congestion thus in order to provide reliability and handle congestion.
- Some existing TCP Congestion Algorithms are TCP Tahoe, TCP Reno, TCP Vegas, TCP Veno, TCP Cubic, TCP BBR etc.

Terminologies used in TCP Congestion Algorithm

In many TCP Congestion Control Algorithms few of the common terminologies used are following:

- CWND (Congestion Window) - Amount of data which can be transmited reliably without an ACK.
- RWND (Receiver Window) - Amount of data that the destination can receive.
- RTT (Round Trip Time) - It is the time required for a packet to travel from a specific source to a specific destination and back again.
- ssThresh - The slow start (SS) threshold (ssthresh) determines the (de)activation of slow start. The SS will be discussed briefly in later slides.

TCP Reno

- TCP Reno treats the occurrence of packet loss as a manifestation of network congestion
- It is based on Sliding windows comprising of :
  - Slow Start (SS)
  - Congestion Avoidance (CA)
  - Fast retransmit
  - Fast recovery

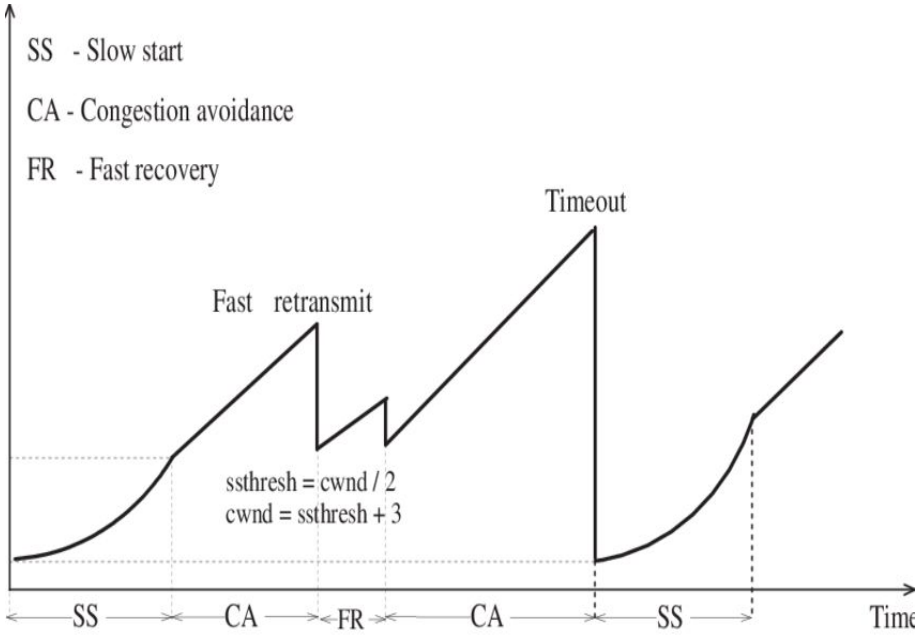


Image Credit : https://bit.ly/3nLj5YQ

TCP Reno : SS (Slow Start - *Exponential Increase*) + Conegetion Avoidance (*Linear Increase*)

- **Slow start** - It is used to initiate a connection.
  - In slow start,
    - cwnd is set to 1 (initialized) segment (1 MSS).
    - With each ack IF CWND < ssThresh,
      - increase CWND by 1 (exponential increase)
  - Aggressive way of building up bandwidth for flow
- **Congestion Avoidance** - when CWND >= ssThresh.
  - In congestion Avoidance Phase (No Congestion),
    - With each ack received,
      - increase CWND by 1/CWND (Linear Increase)
  - Congestion Detected : ssThresh = CWND / 2
  - Timeout : CWND = 1

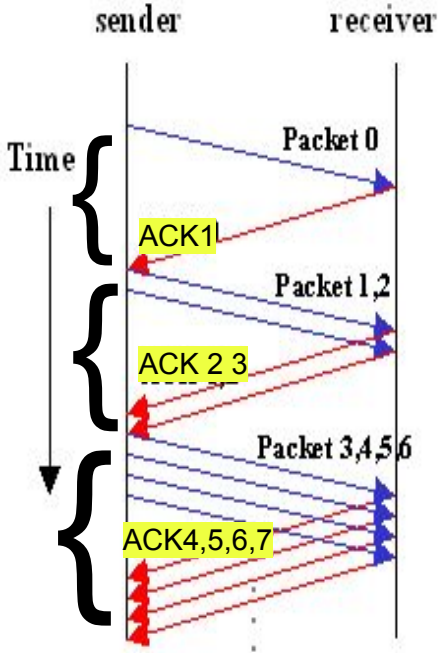


Image Credit : https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/e-slow.gif

TCP Reno : Fast Retransmit and Fast Recovery

- **Fast retransmit and Fast recovery**
  - After three duplicate acks, Reno assume packet loss
    - Retransmit the packet
    - SET ssThresh = (1/2) X CWND,
    - SET CWND = ssThresh + 3
  - For each **duplicate ack**, increment cwnd by **1** ***(keep flow going)***.
  - When **new data acked**, **Exit from Fast Recovery** Mode and do regular congestion avoidance.
    - Thus, after exit SET CWND = ssThresh

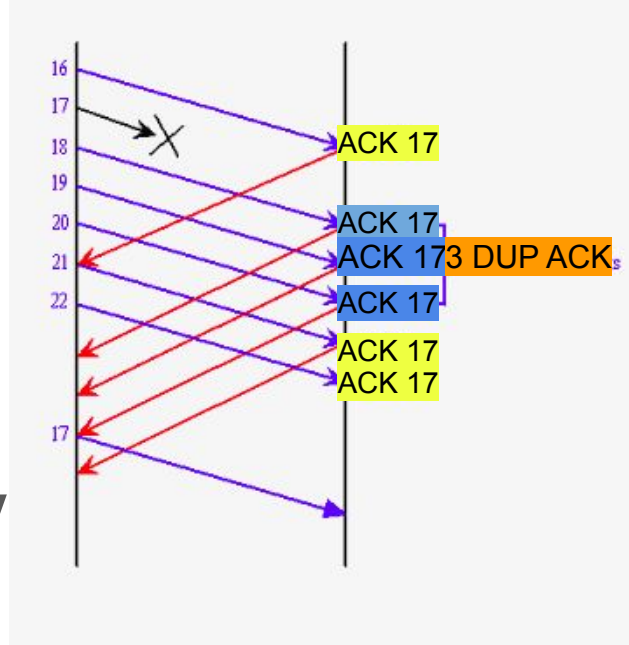


Image Credit: https://bit.ly/3cHAPy7

TCP Reno : Timeout

- Timeout
  - TCP maintains a Retransmission Timer
  - The duration of Timer is called as RTO (Retransmission Timeout)
  - When Timeout occurs :
    - SET ssThresh = CWND / 2
    - SET CWND = 1
  - Algorithms enters into SS phase.

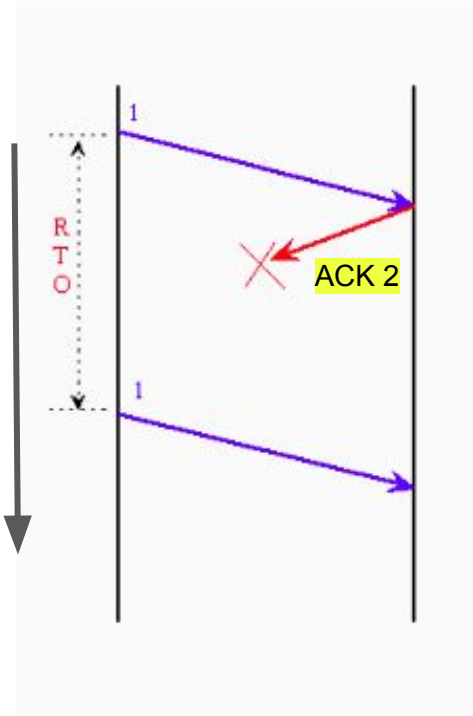
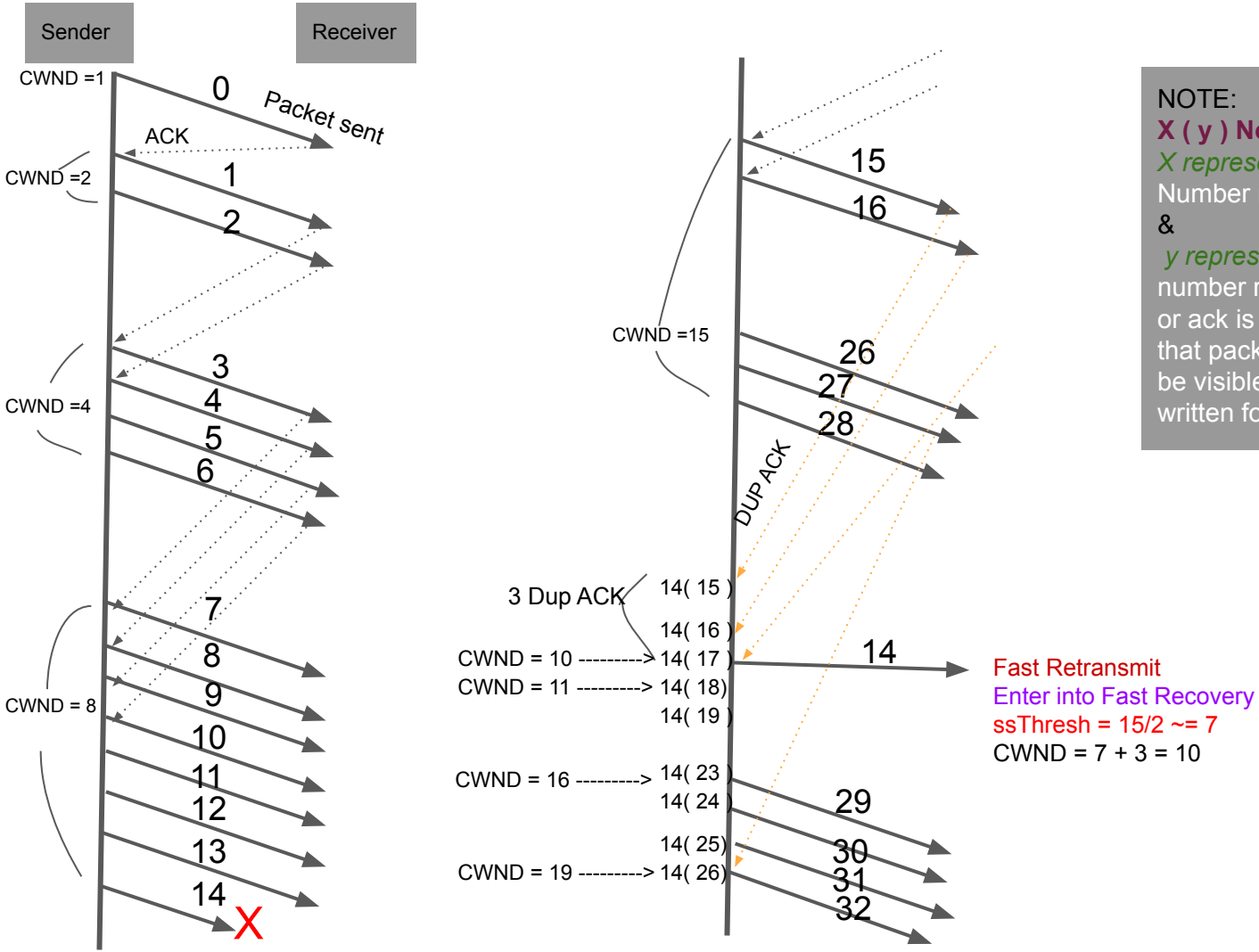


Image Credit : https://bit.ly/3COHfG0

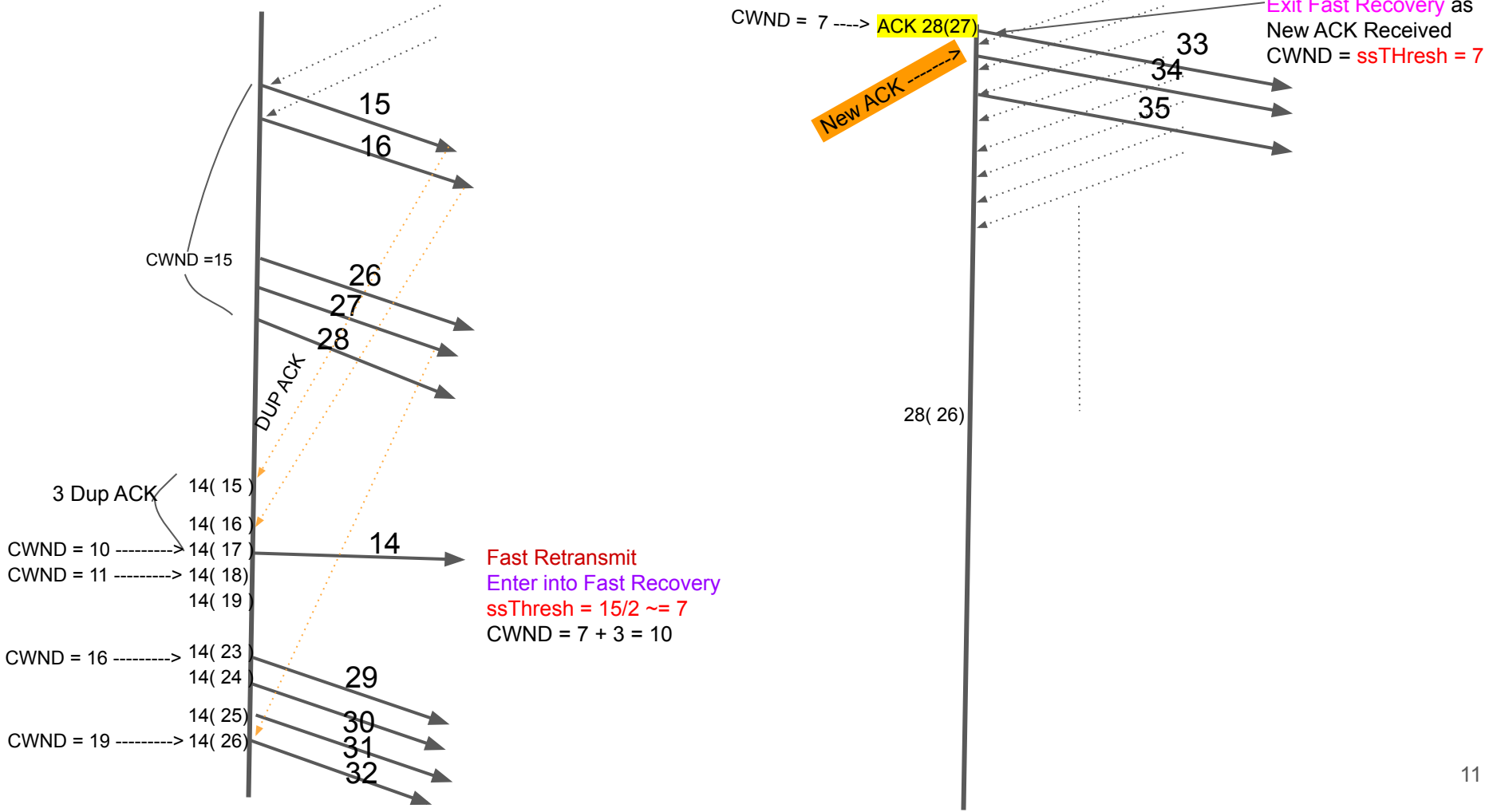
TCP Reno Example

Example is Inspired from :https://bit.ly/3oUDEl3
Note : Example is drawn on Slides Manually.



Fast Retransmit
Enter into Fast Recovery
ssThresh = 15/2 ~= 7
CWND = 7 + 3 = 10

NOTE:
X (y) Notation :
X represents -> The next sequence Number receiver is asking for.
&
y represents -> Shows packet sequence number responsible for generating ack/ or ack is generated as a consequence of that packet sqn number though it won't be visible to us(the sender) it's just written for explaining purposes.


15
16
CWND =15
26
27
28
DUP ACK
3 Dup ACK
14(15)
14(16)
CWND = 10 ---------> 14(17)
CWND = 11 ---------> 14(18)
14(19)
14
Fast Retransmit
Enter into Fast Recovery
ssThresh = 15/2 ~= 7
CWND = 7 + 3 = 10
CWND = 16 ---------> 14(23)
14(24)
14(25)
CWND = 19 ---------> 14(26)
29
30
31
32
CWND = 7 ----> ACK 28(27)
New ACK ------->
33
34
35
Exit Fast Recovery as
New ACK Received
CWND = ssTHresh = 7
28(26)

TCP Vegas

- **BASIC IDEA**
  - **TCP Vegas** emphasizes packet delay, rather than packet loss, as a signal to help determine the rate at which to send packets.
  - Controls the congestion window by measuring the roundtrip times (RTT) of the packets.

- **New Retransmission Mechanism**
  - Decreases **CWND** multiplicatively using a factor of ¾ rather than Tahoe/Reno's ½.

- **Change In Triggering Fast Retransmit**
  - On Receiving dupACK check:

    If CurrentTime - SendingTime + BaseRTT is greater than TimeOutValue ➔

    Then Trigger FastRetransmit without 3 dupACK.

TCP Vegas : Algorithm

- Expected Flow Rate (**Expected**) = CWND/**BaseRTT**
- CurrentFlowRate (**Actual**) = CWND/**RTT**
- **DIFF** = (**Expected - Actual**)

CASE 1 : IF DIFF X BaseRTT lies in range (α , β)

Remain CWND as it is (No change).

CASE 2 : IF DIFF X BaseRTT < α

Increase the CWND by 1.

CASE 3 : IF DIFF X BaseRTT > β

Decrease the CWND by 1.

BaseRTT :
Minimum of all Measured RTT also called as minimumRTT (Continually keep updated)

RTT : Round Trip Time

α , β are threshold set by experimentation. These controls utilization of bandwidth without overloading.

NOTE : This DIFF X BaseRTT is considered as **Extra Data** added to the network.

Main Ideas

Vegas Main Idea

- Maintain the extra data (queue size) in the bottleneck router buffer to be between the lower α and upper threshold β.
- To reach high throughput
- Avoid packet overflow
- Try to find the correct window size without incurring a loss.

One Major Disadvantage of Vegas:

- Performs significantly degrades when coexists with concurrent Tcp Reno Flows, due to its proactiveness.

TCP Veno : (Vegas + Reno)

Main Ideas :

- Distinguishing between congestive & non-congestive (random error) states .
- Use mechanism similar to Vegas to gauge network state (Congestion/Random loss).

Major Changes from existing TCP Reno :

- **Refines the multiplicative decrease** (MD) algorithm of TCP Reno by adjusting the slow-start threshold according to the perceived network congestion level rather than a fixed drop factor of 2.
- **Refines the linear increase algorithm** so that the connection can stay longer in an operating region in which the network bandwidth is fully utilized.

TCP Veno : Algorithm Discussion

- Expected Flow Rate (**Expected**) = CWND/**BaseRTT**
- CurrentFlowRate (**Actual**) = CWND/**RTT**
- **DIFF** = (**Expected - Actual**)

When RTT > BaseRTT there is a bottleneck link where packets accumulates . Let denote backlog at Queue by N.

RTT - BaseRTT = N/**Actual**

N = **DIFF** X **BaseRTT** = Extra Data

If Packet Loss is Detected :

CASE 1 : IF N < β

- **Veno Assumes Loss is Random.**

CASE 2 : IF N >= β

- **Veno Assumes Loss is Congestion.**

BaseRTT :
Minimum of all Measured RTT also called as minimumRTT (Continually keep updated)

RTT : Round Trip Time

α , β are threshold set by experimentation. β = 3 works best

TCP Veno : Refining Additive Increase

- Additive increase algorithm
- **Goal** : let connection can stay in operating region longer

IF N < β // available bandwidth under-utilized

cwnd=cwnd+1/cwnd when every new ack received.

IF N >= β // available bandwidth fully utilized

cwnd=cwnd+1/cwnd when every other new ack received.

TCP Veno : Refining Multiplicative Decrease

What Reno will do (Reno Fast Recovery)

1). Retransmit missing packet

Set **ssthresh = cwnd / 2**

Set cwnd = ssthresh + 3

2) Each Time another dup ACK arrives increment cwnd by 1

3) When new ACK arrives sett cwnd to ssthresh and exit .

Veno modify Step 1 of Reno

IF DIFF X BaseRTT < β // random loss due to bit errors is most likely to have occurred

ssthresh = $cwnd_{loss}$ X (4/5)

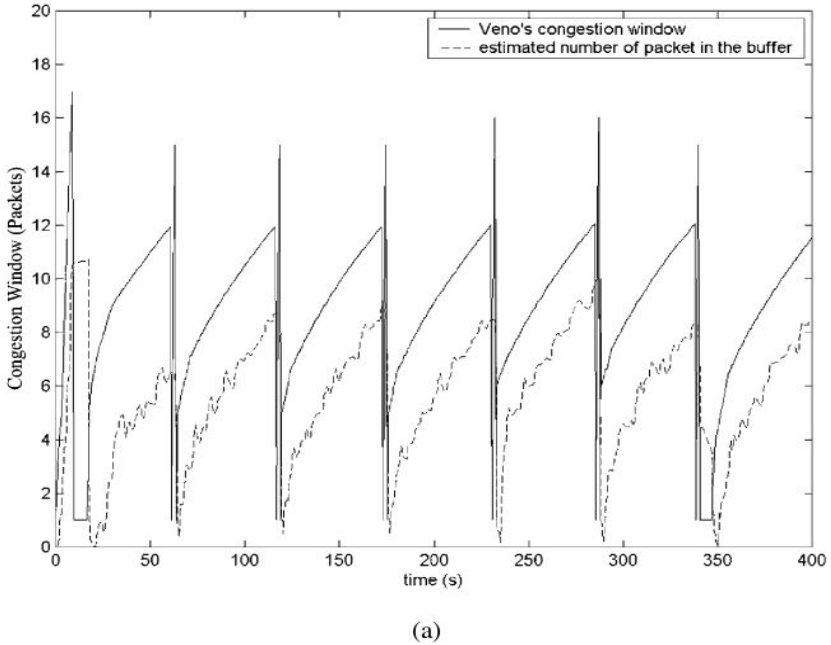
ELSE // congestive loss is most likely to have occurred

ssthresh = $cwnd_{loss}$ / 2

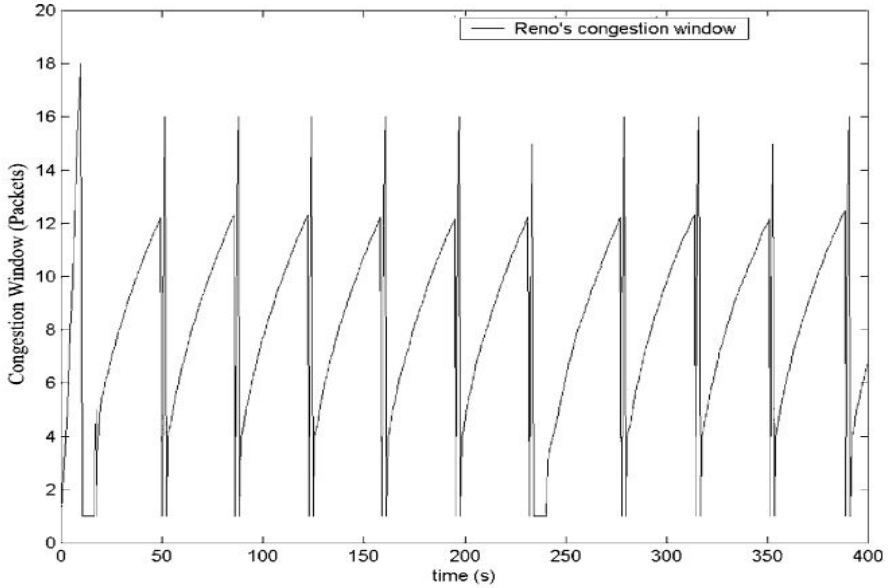
This factor must lie between 0.5 and 1 and gives better result when > 0.75.

TCP Veno and Reno Window Evolution Graph

(When there is No Random Loss.)



(a)



(b)

a) TCP Veno Window Evolution

b) TCP Reno Window Evolution

Both Images Credits : https://ieeexplore.ieee.org/document/1177186

TCP Veno : Summary

- In summary, veno only refines the additive increase , multiplicative decrease (AIMD)
- All other parts of Reno remain intack Including slow start, fast retransmit, fast recovery, computation of retransmission timeout algorithm

References:

https://en.wikipedia.org/wiki/TCP_congestion_control

https://www.geeksforgeeks.org/tcp-congestion-control/

https://developer.mozilla.org/en-US/docs/Glossary/TCP_slow_start

https://ieeexplore.ieee.org/document/1177186

https://www.powershow.com/viewht/108eed-ZDc1Z/TCP_Veno_TCP_Enhancement_for_Transmission_Over_Wireless_Access_Networks_powerpoint_ppt_presentation