CSE 588 May 7th Notes

Intuition in understaning how TCP works

```
Sender   --------------------------------Router------------------------------------Receiver
           Latency=1                 buffers                 latency =0
          bw = oo                                                         bw=1
```

Assume the receiver has infinite buffer.
Window size increases by 1 each time a packet is recieved successfully. To keep pipe full, there need 4 or 5 packets (4 in the router buffer and 1 in the receiver).

Sender doesn't realize loss of packet for a while even after the pipe is full and sender packets are dropped at router.

OPTION 1: while not have equal bandwidth: can't slow every link to the slowest link speed. Or do circuit switch and reserve circuit and set the bandwidthes to the speed of the accessor.
OPTION 2: While not have more buffer

Is it a good idea for the router to alert the sender?
It might work in this case.
Why not drop the first packet, rather the last one?
It is not going to make a big difference. Also router buffer usually is a FIFO queue and it is odd to drop the first packet.

When sender gets the out of order packet, it stops sending. Then it cut the window size to half of what it was before. And it eventually figures out that it needs to do slow starts again and resends the packets.

Try congestion avoidance:
Send packets until 11th packet, flag it and only after it comes back to the sender, the sender can increase the window size by 1 again. If too many packets are sent, one packet is dropped, the good thing is the window size stays the same. So only one packet will be lost.
After sender resends the lost packet, it does fast recovery by increase the window size by one everytime a packet is ACKed by the receiver.

WINDOW size figure

Increase physical resources (buffer size) actually hurts the network because more packets will be lost before the loss is detected.
If the buffer size is too small, then the network can't reach the prime performance.

Router assist

ECN annote the packets
- queue size
- avg queue size
- if avg >x: RED with ECN

Better end host SW

Vegas: use the signal of the round trip time of the packets to estimate the queue size and backoff if >1.

If the packets come backat the same rate even if the sender sends more packets out at the same time, that means it doesn't need to increase the window size.
Challenge: fairness. If Vegas shares with TCP, then Vegas backs off faster than TCP and it is not fair.

If there are several senders share the resources, the throughput can drop to less than 1 for each sender.

If the bandwidth is very big,
figiure

TCP pacing(it works well in the case of little buffer)
Instead of sending the packets immediate after a previous packet is acked, the sender paces the sending of the packets.   However it doesn't work well in general because the network will be full suddenly for everyone and everyone will need to backoff at the same time and they are synchronized. TCP pacing performs worse than TCP.

8888888888888888888888888888888888888888888888888888888888888888888888888888888
Short connections
    (a) there is no congestion, everyone learns the bandwidth individually
    (b) congestion: congestion signal is set randomly on someone.   No one else backs off.
TCP friendliness
Should we start at 1 packet or 4?

Active queue management
    -RED (Radom ..) RED is easy to implement
        i)avoid synchronization
        ii) large buffer and low latency

Penalty Box
Monitor the top senders and do RED on these senders rather on the slow senders.

Complexity of hardware
FIFO simple
Non FIFO queue is harder

RED is not too hard
Fair queueing with monitoring of per flow state: each queue for each sender. It is harder.
ATM with monitoring of per flow state.
When to signal? avg buffer > BW*RTT OR rate of increase
Any fixed constant doesn't work for RED.