BITS Pilani
Pilani Campus

Computer Networks (CS F303)

Virendra Singh Shekhawat
Department of Computer Science and Information Systems

innovate achieve lead

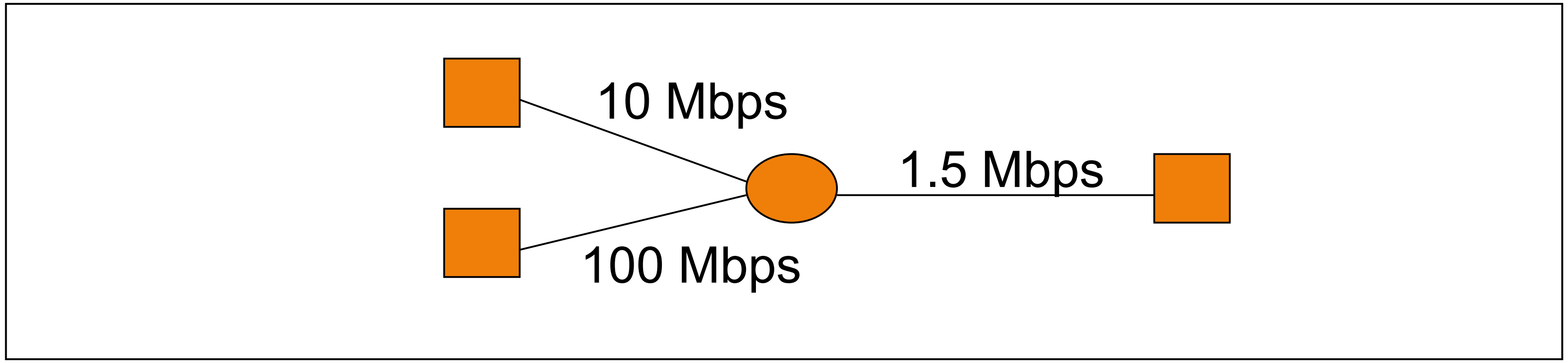
BITS Pilani
Pilani Campus

Second Semester 2020-2021
Module-3 <Transport layer>
Congestion Control

Topics

- Transport Layer
- TCP Protocol
  - Connection Establishment
  - TCP Segment Structure
  - Reliable data transfer
  - Flow control
  - Timeout Estimation
  - **Congestion control**

What is Congestion…?



- Why is it a problem?
  - Different sources compete for resources inside network
  - Sources are unaware of current state of resource
  - Sources are unaware of each other
  - In many situations will result in < 1.5 Mbps of throughput (congestion collapse)

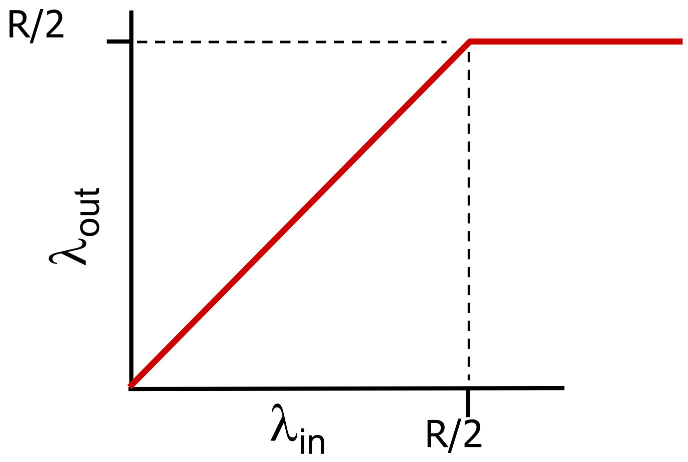
Congestion Control

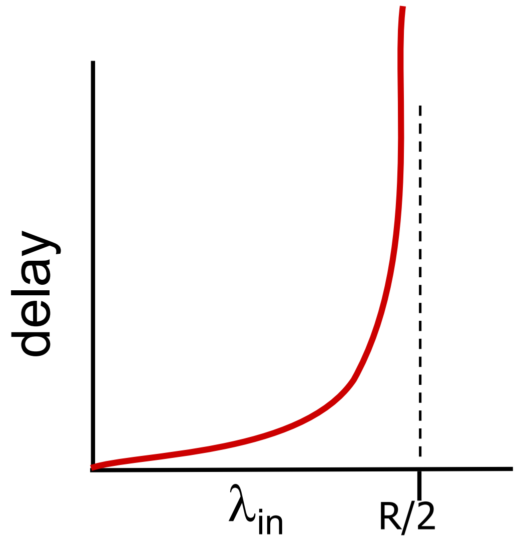
- What is congestion
  - Too many sources sending too much data too fast for network to handle

- Congestion results in
  - Packet losses
  - Packet delays
  - Throughput reduction

Causes/Cost of Congestion [.1]

- Two sender and two receivers
- One router with infinite buffers
- Output link capacity R
- No retransmission





Causes/Cost of Congestion [..2]

- One router, *finite* buffers
- Sender retransmission of timed-out packet
  - Application-layer input = Application-layer output: $\lambda_{in} = \lambda_{out}$

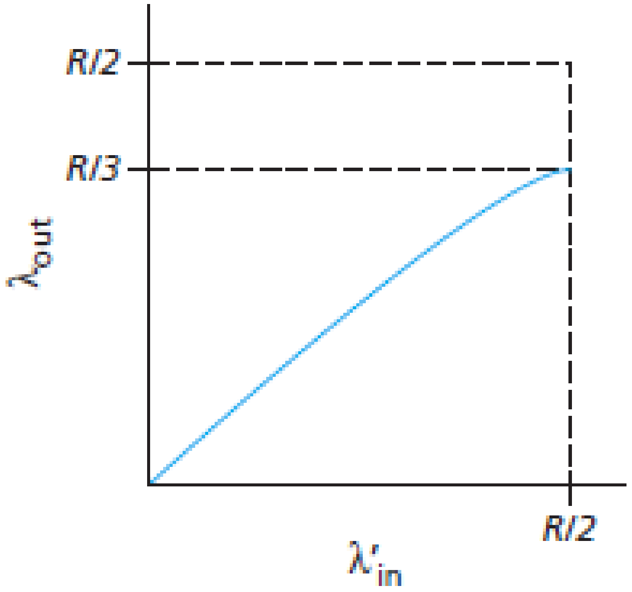
  - Transport-layer input includes *retransmissions*:

    $\lambda_{in} > \lambda_{in}$

Causes/Cost of Congestion […3]

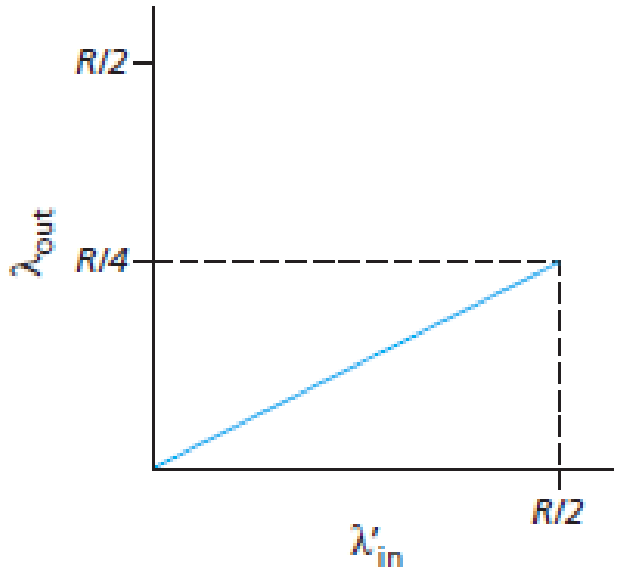
- Packets can be dropped at router due to full buffers
- Sender only resends if packet known to be lost (Tricky Assumption…)

- Cost of congestion
  - Retransmission for dropped packets

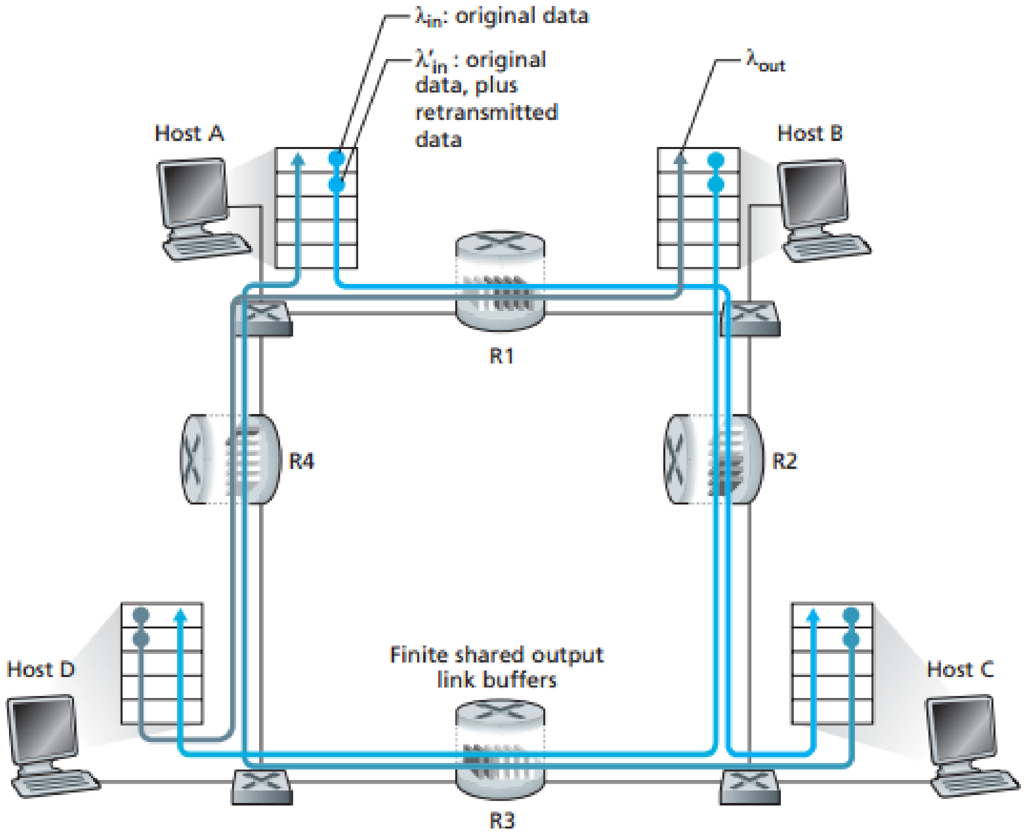


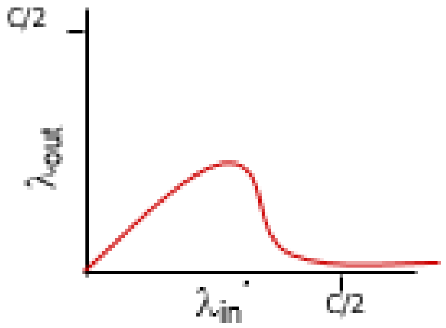
Causes/Cost of Congestion [....4]

- Packets can be lost, dropped at router due to full buffers
  - Sender times out prematurely, sending *two* copies, both of which are delivered



Causes/Cost of Congestion [.....5]





When packet dropped, any "upstream transmission capacity used for that packet was wasted!

Approaches Towards Congestion Control

- Network Assisted Congestion control
  - Routers provide feedback to end systems

- End-to-end Congestion control
  - No explicit feedback from network

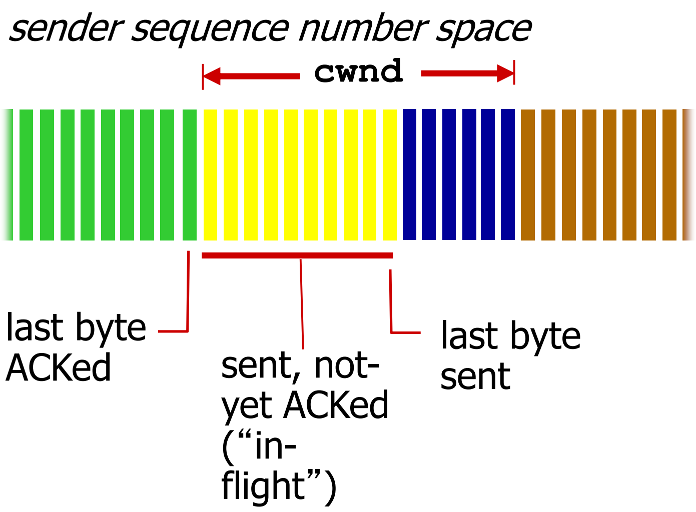
TCP Congestion Control

- Approach
  - Sending rate is a function of perceived congestion

- Arises three important questions
  - How does sender perceive the congestion on the path?
  - What algorithm should be used to change its sending rate?
  - How does sender limit the sending rate?

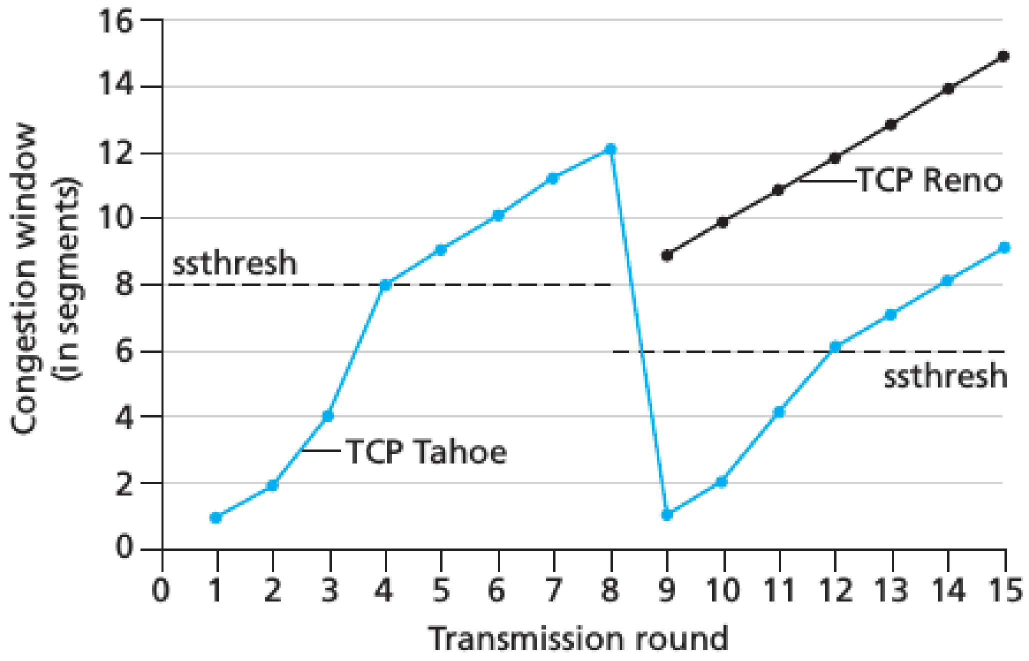
TCP Congestion Control

- Sender limits transmission
  - LastByteSent – LastByteAcked <= min(cwnd, rwnd)



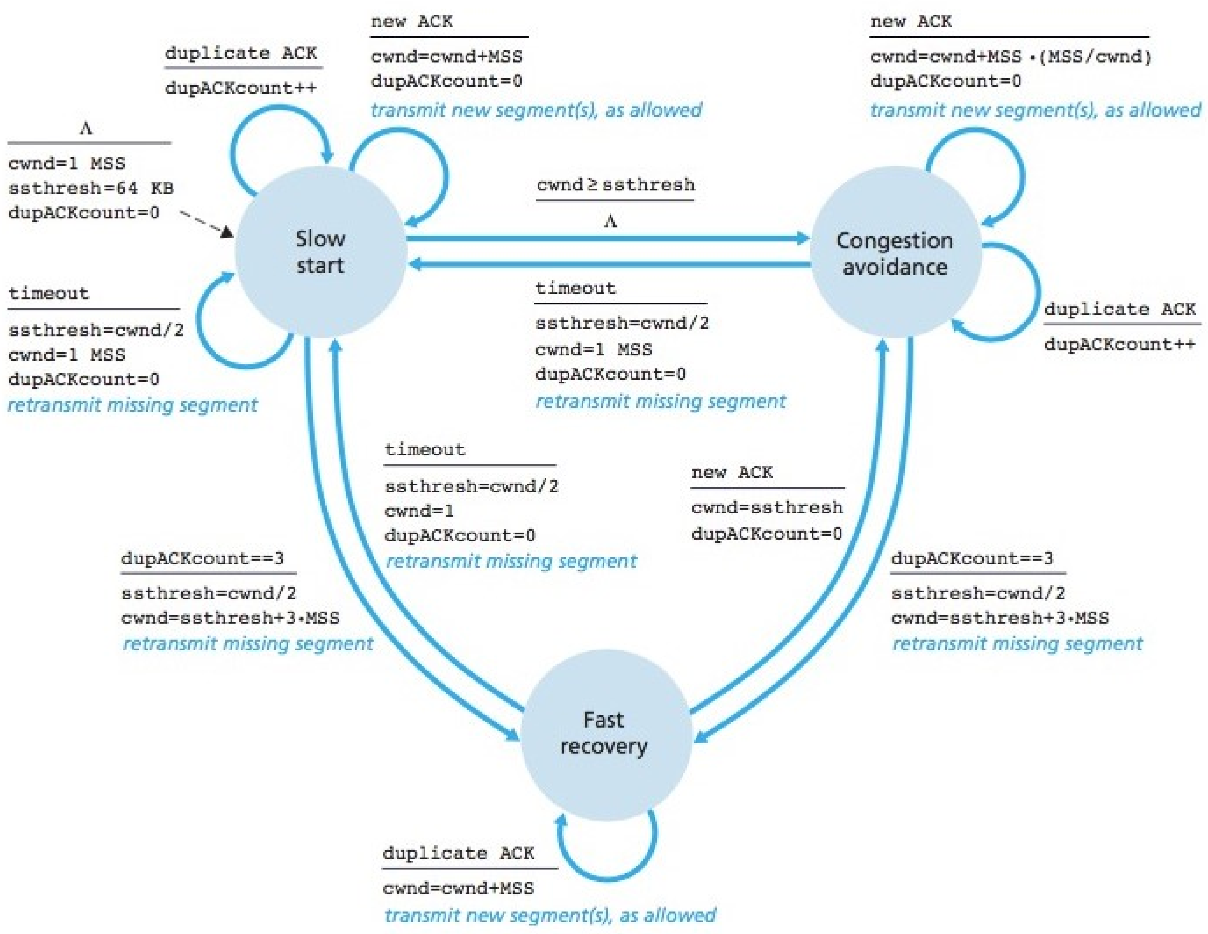
- What is TCP Sending Rate/Throughput?

Switching from Slow Start to CA



Evolution of TCP's congestion window (Tahoe and Reno)

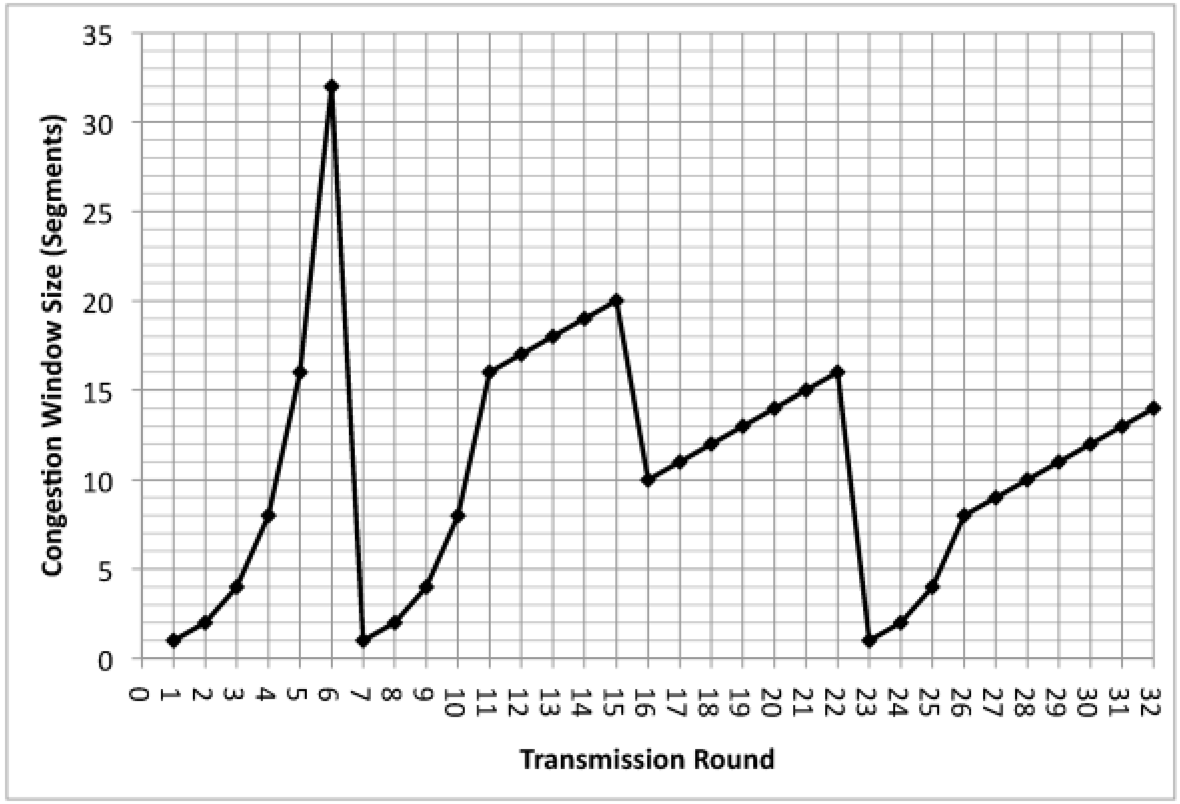
FSM Description of TCP Congestion Control



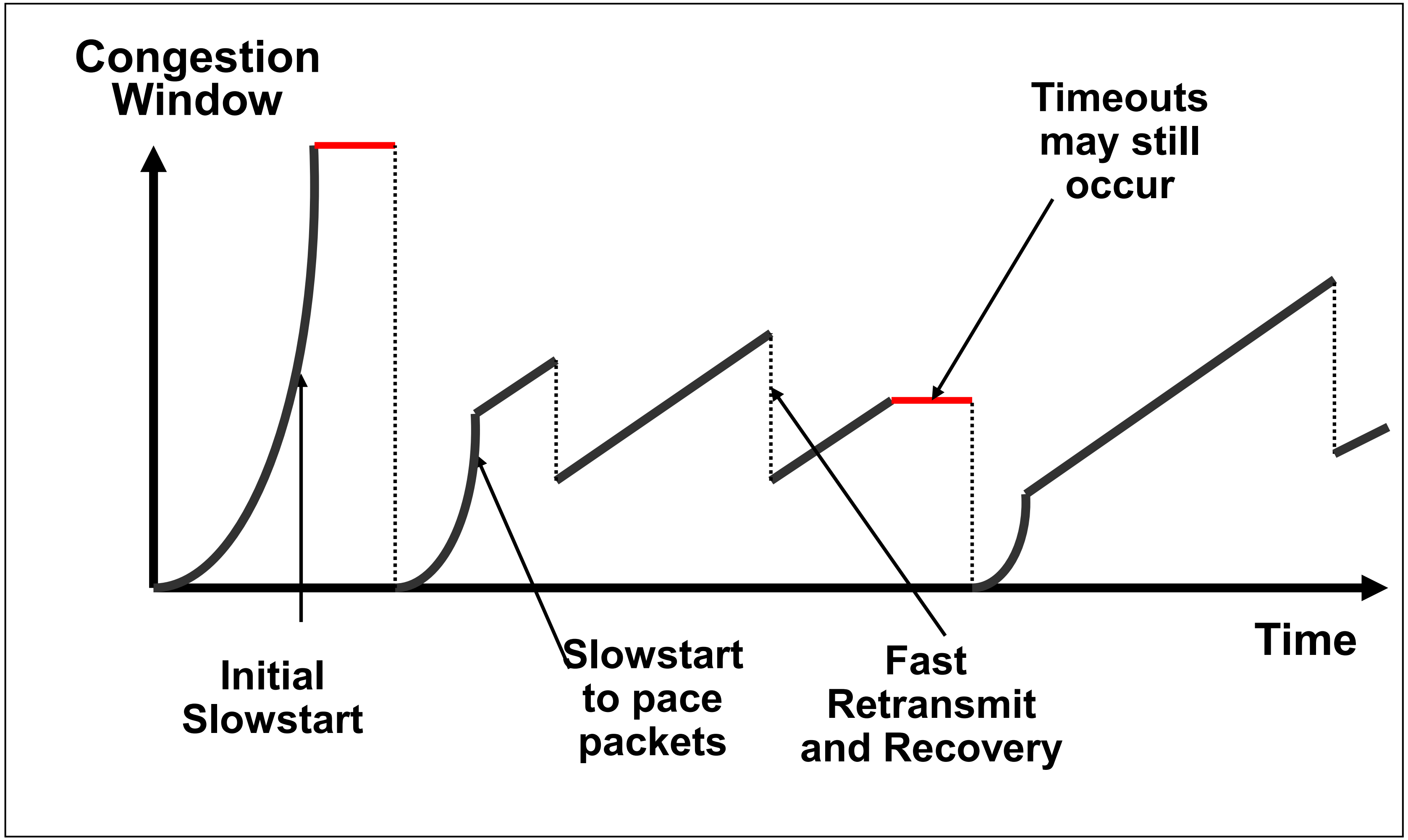
FSM description of TCP congestion control

Example: TCP Congestion Control

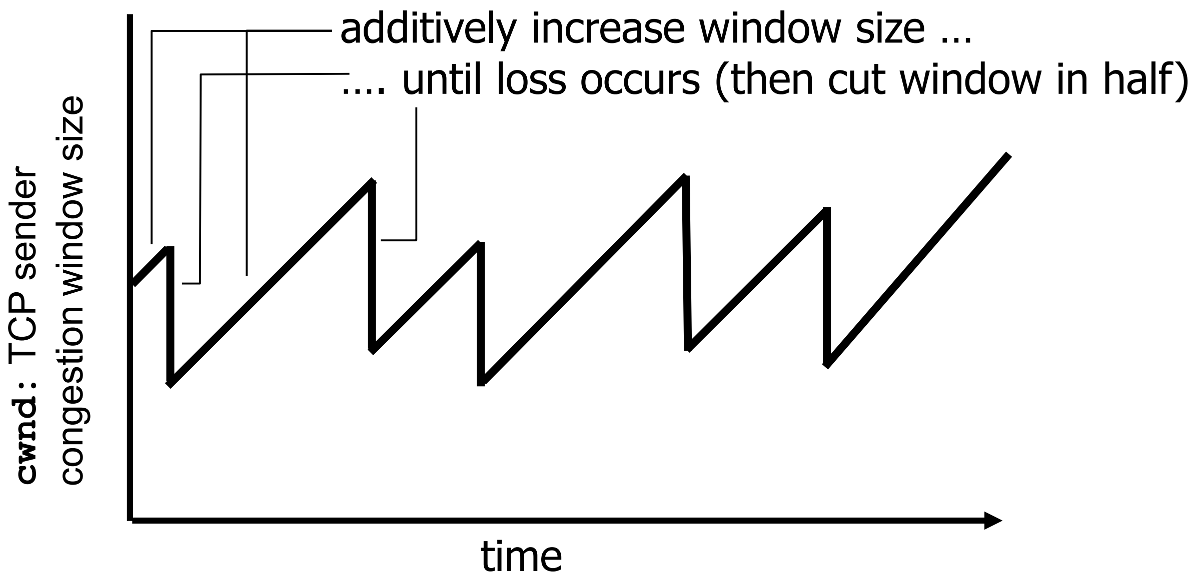
a) Identify the intervals of time when TCP **slow start** is operating.
b) Identify the intervals of time when TCP **congestion avoidance** is operating.
c) What is the **ssthresh** value between transmission round 7-10?
d) What is the **congestion window** value at transmission round 11?
e) How many segments have been sent till transmission round 11? (including 11^{th} transmission round)
f) Identify the intervals of time when TCP **fast retransmission** and **fast recovery** is used?



TCP Sawtooth Behavior



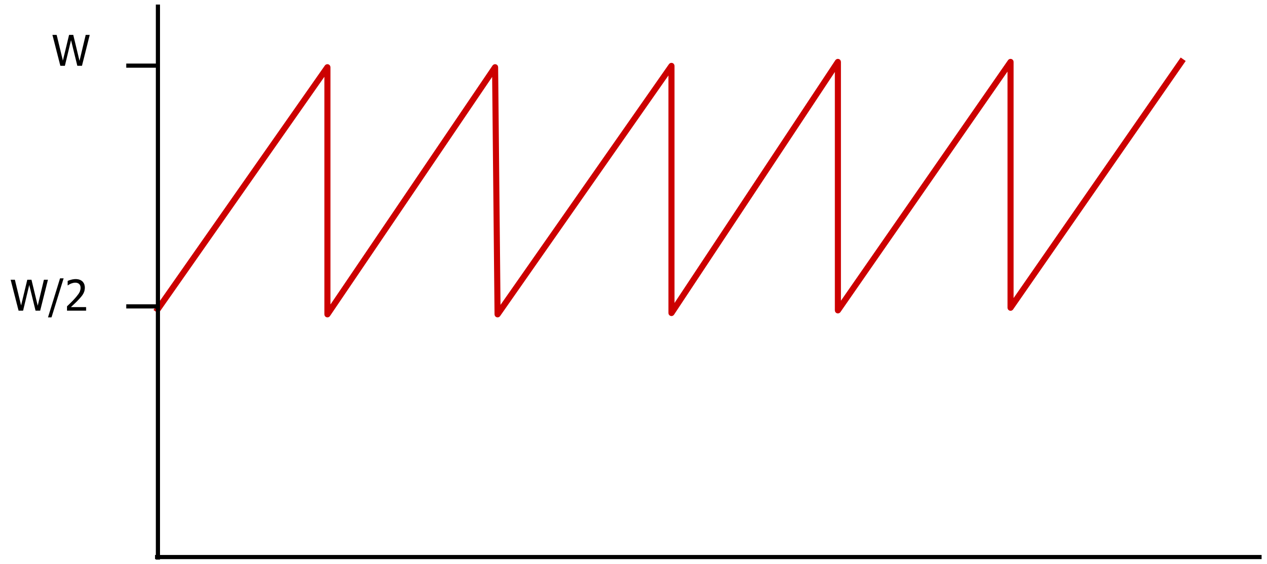
TCP: AIMD Behavior



AIMD saw tooth behavior: probing for bandwidth

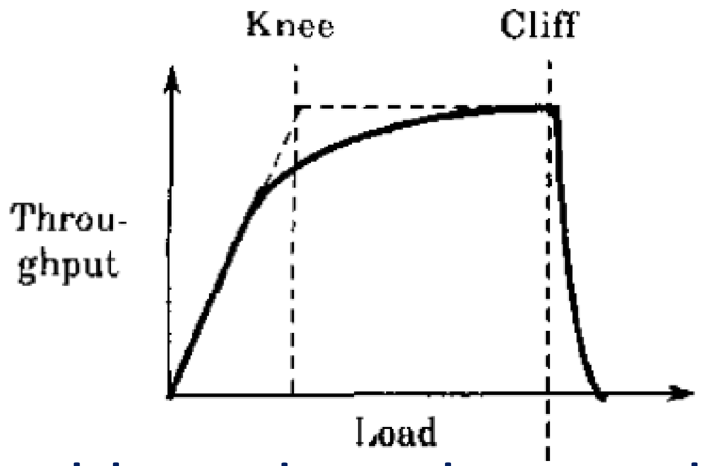
TCP Throughput

- TCP average throughput as function of window size and RTT?



Congestion Control Objectives

- Key to congestion avoidance is the "control function" used to increase or decrease their sending window
  - Distributedness
  - Efficiency: $X_{knee} = \Sigma x_i(t)$
  - Fairness: $(\Sigma x_i)^2/n(\Sigma x_i^2)$



  - Convergence: control system must be stable and reach to goal state from any starting state quickly

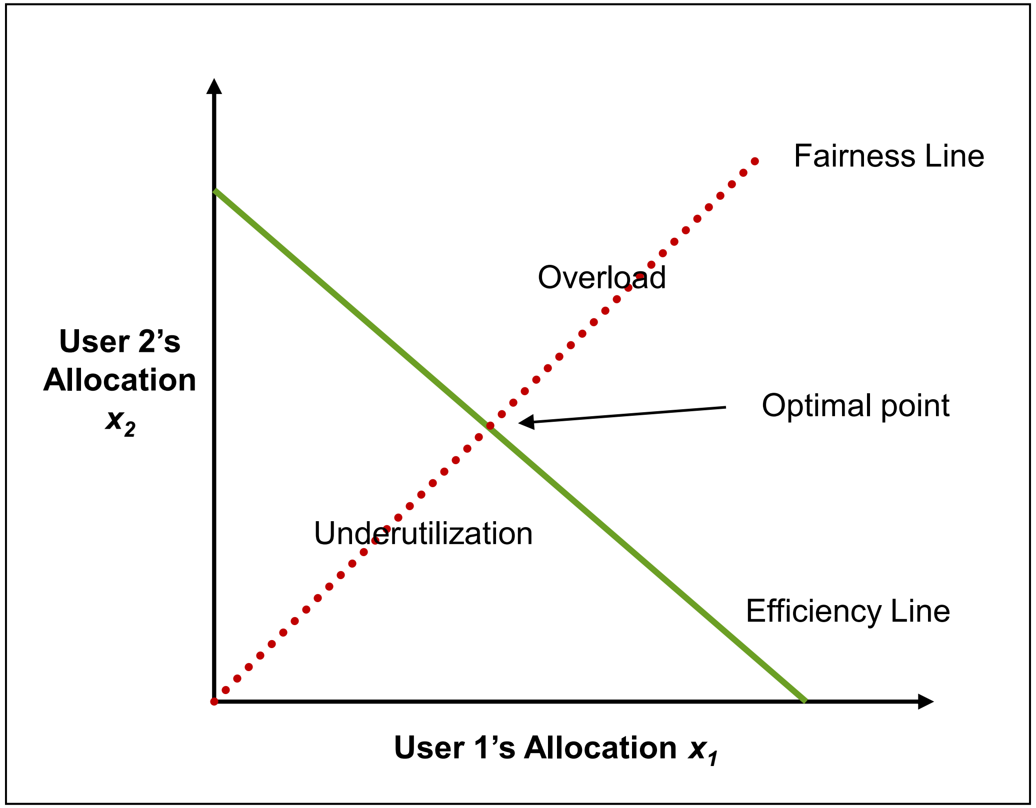
Linear Control

- Many different possibilities for reaction to congestion and probing
  - Examine simple linear controls
  - $W(t + 1) = a + b*W(t)$
  - Different a_i/b_i for increase and a_d/b_d for decrease

- Supports various reaction to signals
  - Increase/decrease additively
  - Increased/decrease multiplicatively
  - Which of the four combinations is optimal?

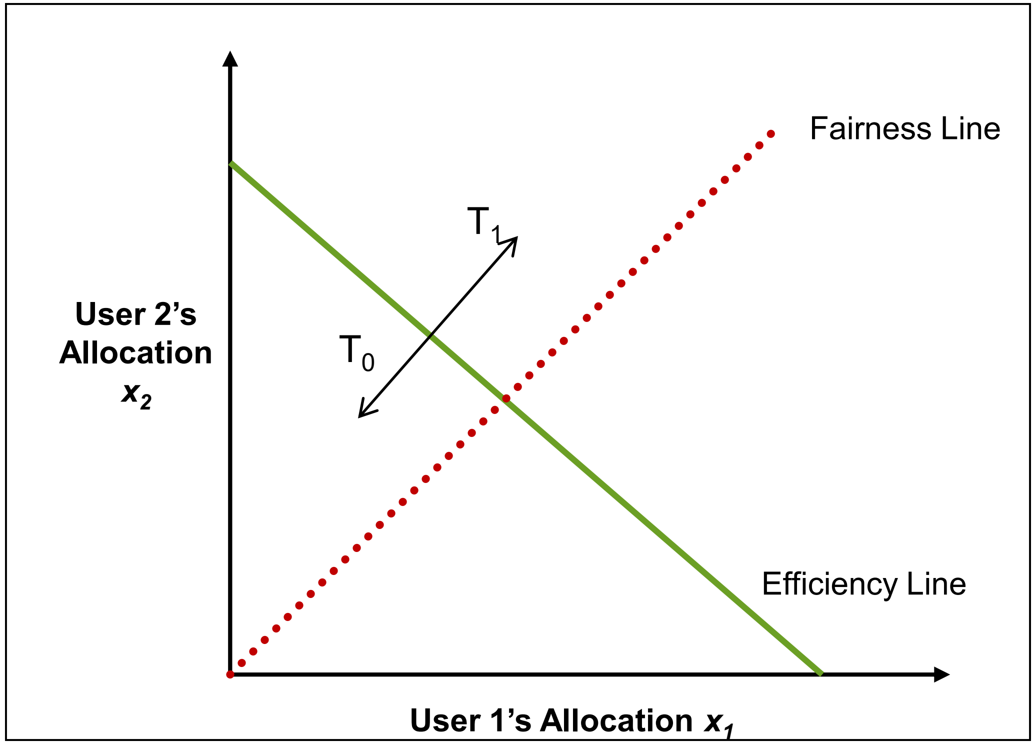
Phase plots (Vector Representation)

- What are desirable properties?



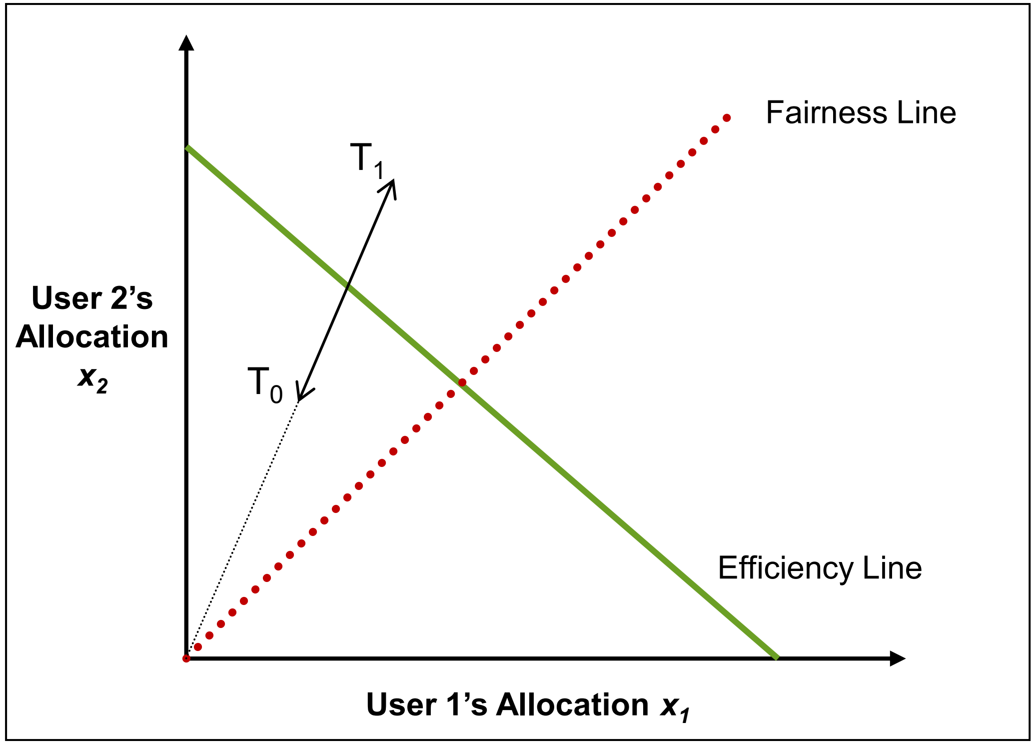
Additive Increase/Decrease

- Both X_1 and X_2 increase/decrease by the same amount over time
  - The additive increase/decrease policy of increasing both users' allocations by a_I corresponds to moving along a 45^0 line



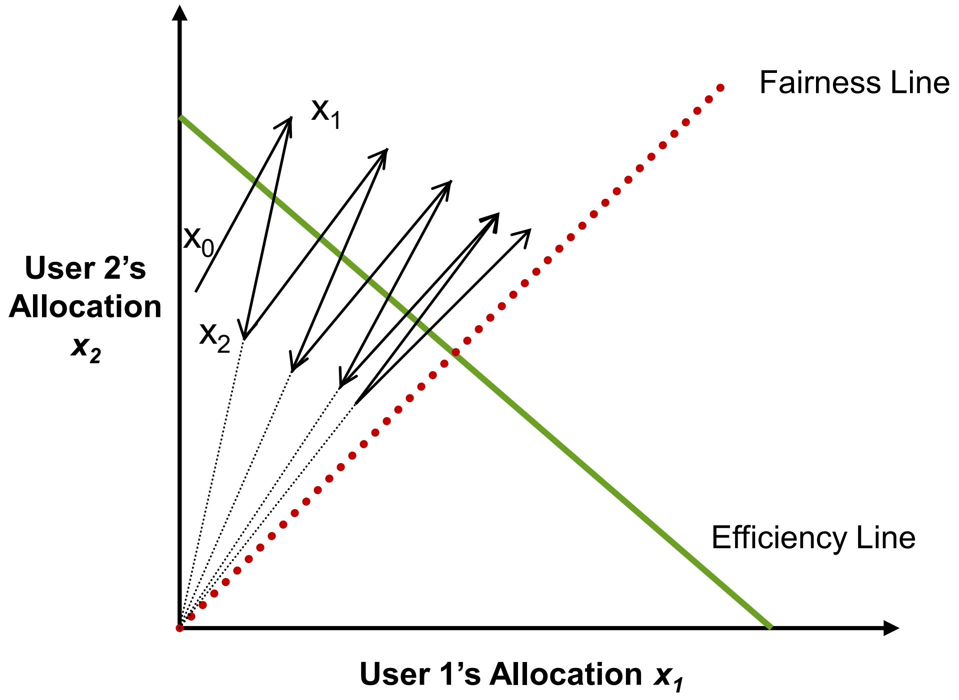
Multiplicative Increase/Decrease

- Both X_1 and X_2 increase/decrease by the same factor over time
  - Extension from origin – constant fairness



What is the Right Choice?

- Constraints limit us to AIMD
  - AIMD moves towards optimal point



TCP Modeling

- Given the congestion control behavior of TCP can we predict what type of performance we should get?
- Important factors which affect the performance:-
  - Loss rate
    - Affects how often window is reduced
  - RTT
    - Affects increase rate and relates BW to window
  - RTO
    - Affects performance during loss recovery
  - MSS
    - Affects increase rate

Simple TCP Model

- Some assumptions
  - Fixed RTT
  - No delayed ACKs

- In steady state, TCP losses packet each time window reaches W packets
  - Window drops to W/2 packets
  - In each RTT window increases by 1 packet, so “W/2 * RTT” before next loss
  - BW(Throughput) = (MSS * avg. window)/RTT

Simple Loss Model

- What is the loss rate?
  - Segments transferred = (0.75 W/RTT) * (W/2 * RTT) = $3W^2/8$
  - Loss rate = L = $8/3W^2$
  - W = sqrt(8/3L)
- BW = 0.75 * MSS * W / RTT
  - BW = MSS / (RTT * sqrt (2*L/3))

TCP Performance

- Example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- Throughput in terms of loss rate:

$$Throughput = \frac{1.22 \times MSS}{RTT\sqrt{L}}$$

- $L = 2 \times 10^{-10}$
- Requires average congestion window as 83,333 segments
- To get 10 Gbps throughput, it can afford one loss event for every 5,000,000,000 segments

Silly Window Syndrome[.1]

- Should the sender transmit a half-MSS or wait for the window to open to a full-MSS?
  - Early implementations of TCP allows transmission of half MSS
  - This strategy can lead to **silly window syndrome**

- What is silly window syndrome?
  - Either sender transmits a small segment or the receiver opens the window a small amount

Silly Window Syndrome[..2]

- It is not possible to outlaw sending small segments. Why?

- But we can keep the receiver from introducing small “containers”
  - After advertising zero window, receiver must wait for space equal to an MSS before it advertises again
  - Receiver can delay ACKs and sends combined ACK
    - It has no clue how long should wait !!!

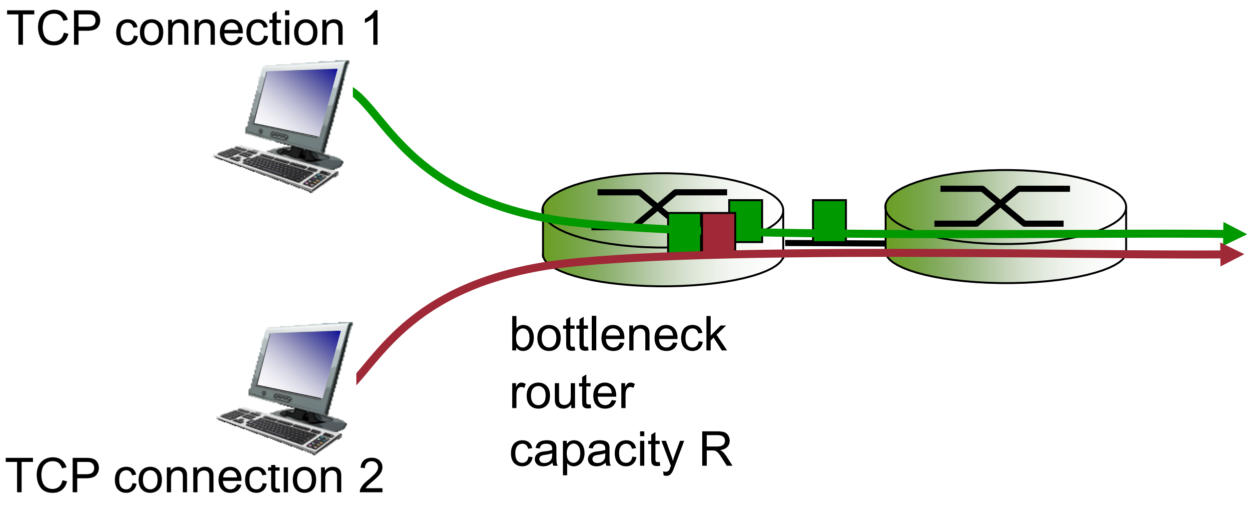
Nagle's Algorithm

- When does the TCP sender decide to transmit a segment?
  - How long sender should wait?

```
When the app produces data to send
    if both the available data and window >= MSS
            send a full segment
    else
        if there is unACKed data in flight
            buffer the new data until an ACK arrives
        else
            send all the new data now
```

TCP Fairness

Fairness Goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Fairness and UDP

- Multimedia apps often do not use TCP
  - Do not want rate throttled by congestion control
- Instead use UDP
  - Send audio/video at constant rate, tolerate packet loss
- Multimedia applications running over UDP are not being fair. Why???

Fairness and Parallel TCP Connections

- Web browsers often use multiple parallel TCP connections
  – To transfer multiple objects within a Web page

- Application level fairness with multiple parallel TCP connections???

TCP Fairness with different RTT

- Flows sharing bottleneck link with different RTT do not get same bandwidth. Why???
  - BW is proportional to 1/RTT

Compound TCP [.1]

- Compound TCP(CTCP) is a Microsoft algorithm that was introduced as part of the Windows Vista and Windows Server 2008 TCP stack.

- Designed to aggressively adjust the sender's congestion window
  - Focuses on connections with large "**bandwidth-delay**" products
  - Make congestion decisions that reduces the transmission rate based on RTT variations

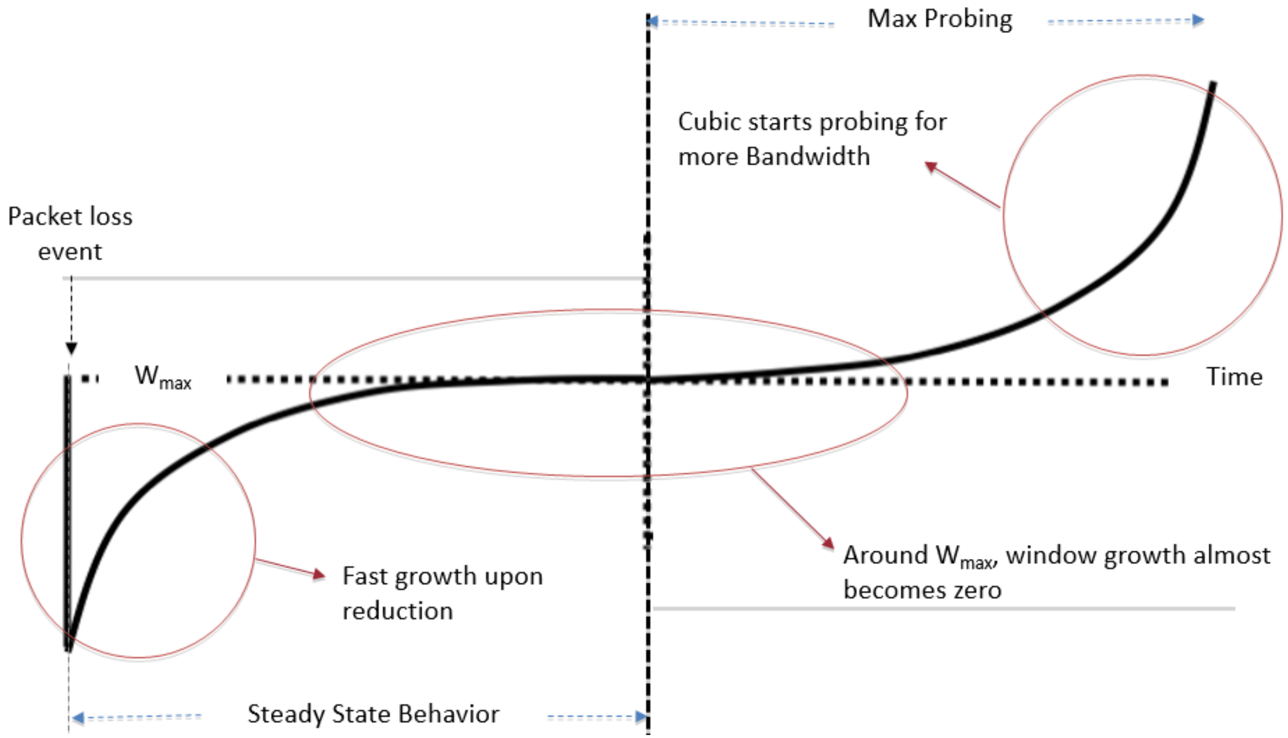
Compound TCP [..2]

- A scalable delay-based component is considered into the standard TCP congestion avoidance algorithm
  - *Sender's win = min (cwnd + dwnd, awnd)*

TCP Cubic

- cwnd = $C(t - K)^3 + W_{max}$
  - W_{max} = cwnd before last reduction
  - β multiplicative decrease factor
  - C scaling factor
  - t is the time elapsed since last window reduction

$$K = \sqrt[3]{W\beta / C}$$



What's Next...

- Network Layer
  - Network layer service models (Internet and ATM)
  - Forwarding versus Routing
  - How a router works?
  - IPv4 Datagram and Fragmentation
  - IPv4 Addressing
    - Hierarchical Addressing
    - NAT, Sub Netting, IPv4 to IPv6 translation, ICMP
  - Routing Algorithms and Protocols
    - Inter-domain Routing and Intra-domain routing
  - Multicast Routing

Thank You