CSC4200/5200 – COMPUTER NETWORKING

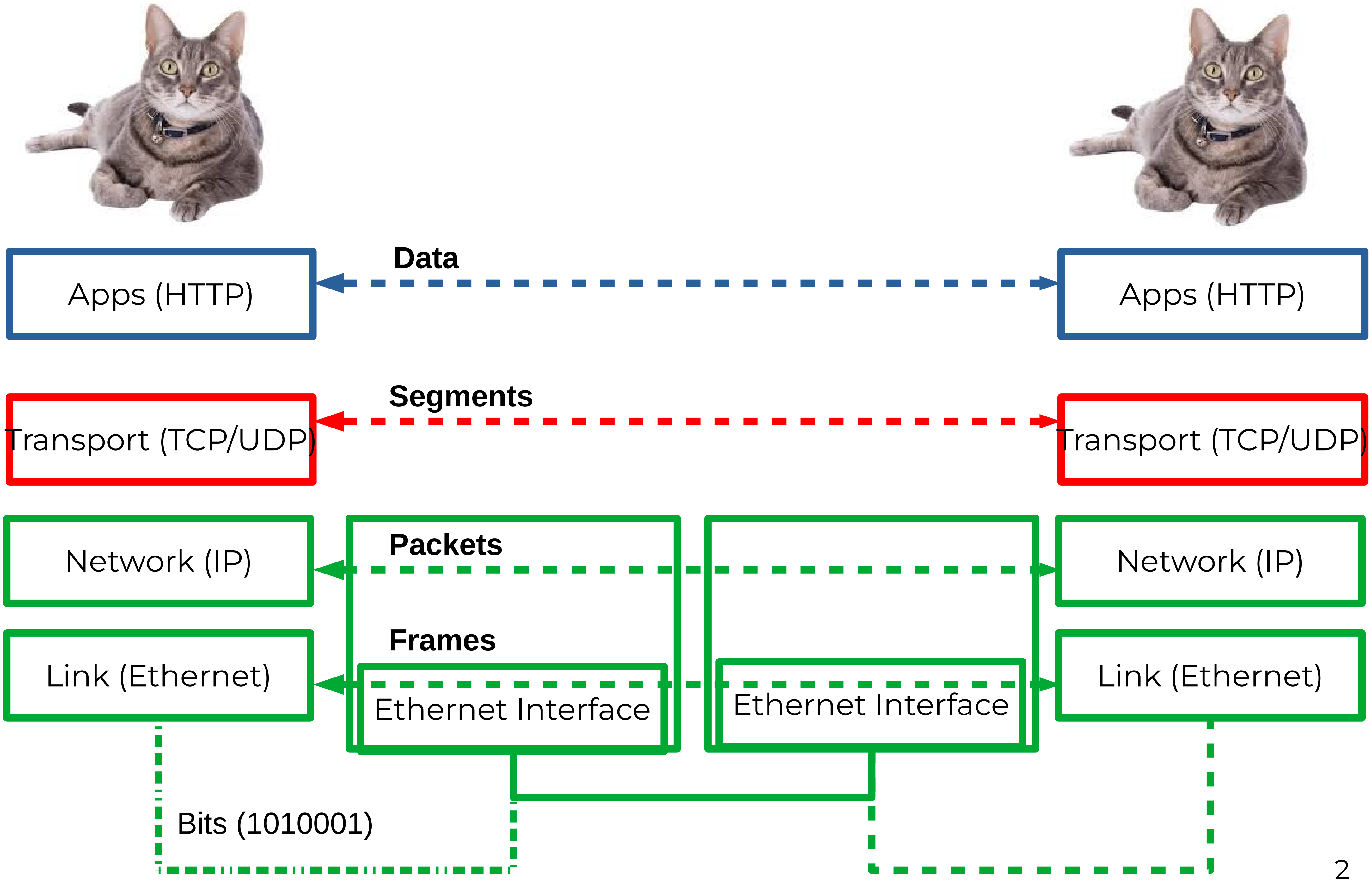
Instructor: Susmit Shannigrahi

CONGESTION CONTROL

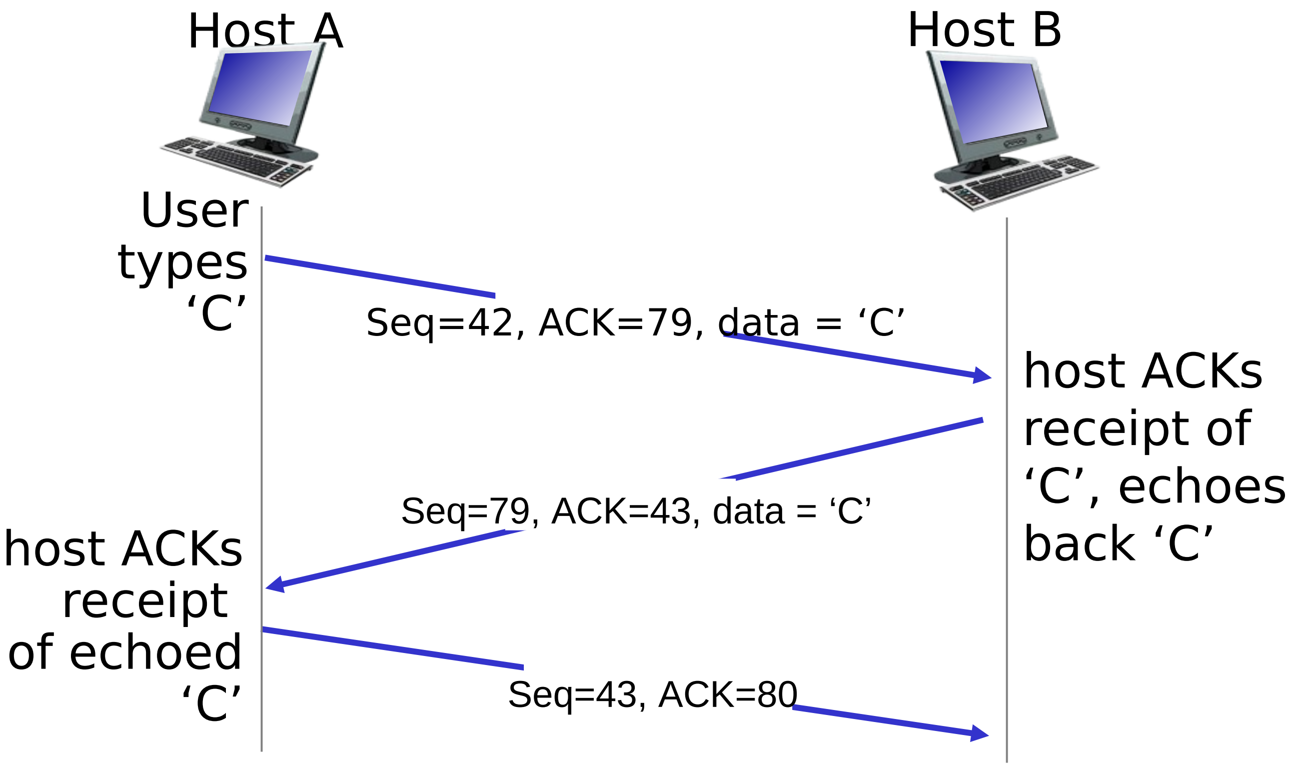
sshannigrahi@tntech.edu

GTA: dereddick42@students.tntech.edu

Tennessee TECH


Data
Apps (HTTP)
Apps (HTTP)
Segments
Transport (TCP/UDP)
Transport (TCP/UDP)
Packets
Network (IP)
Network (IP)
Frames
Link (Ethernet)
Ethernet Interface
Ethernet Interface
Link (Ethernet)
Bits (1010001)

TCP seq. numbers, ISNs



simple telnet scenario

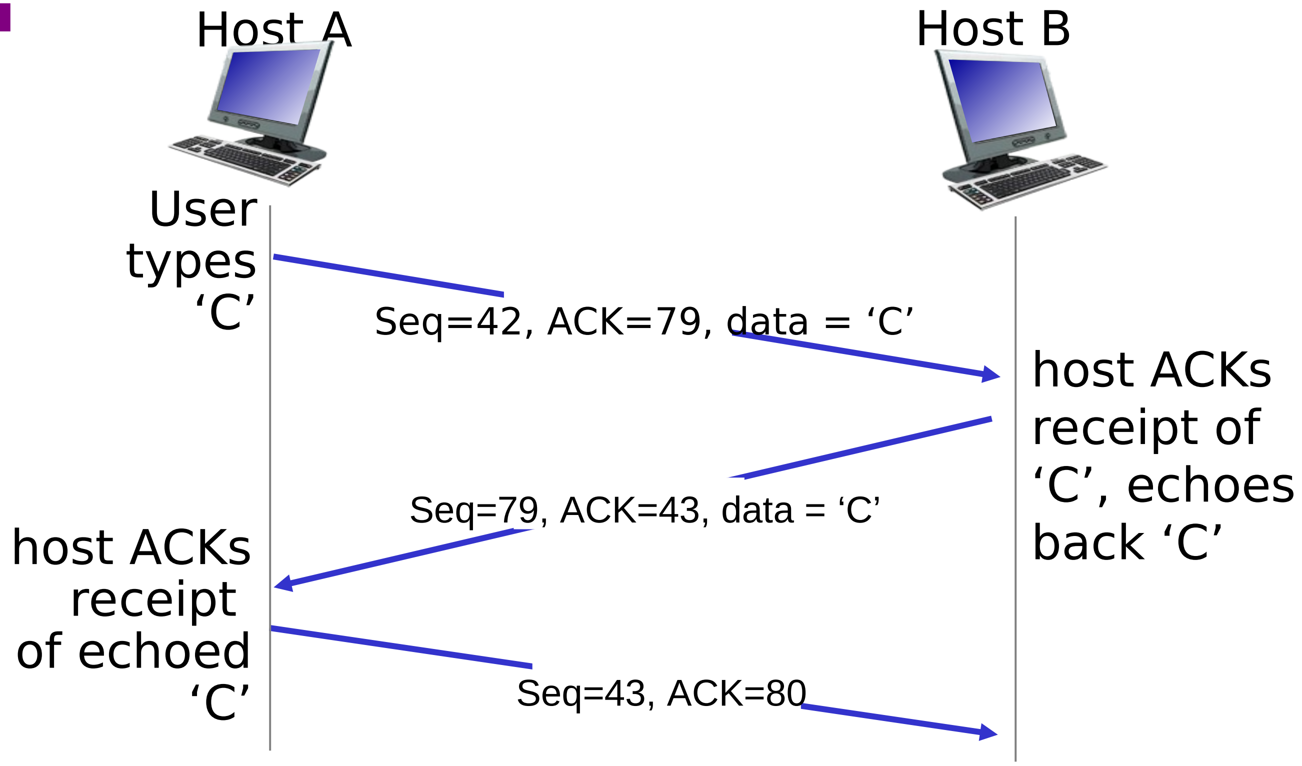
Sequence number for the first byte

Why not use 0 all the time?

- Security
- Port are reused, you might end up using someone else's previous connection
- Phone number analogy

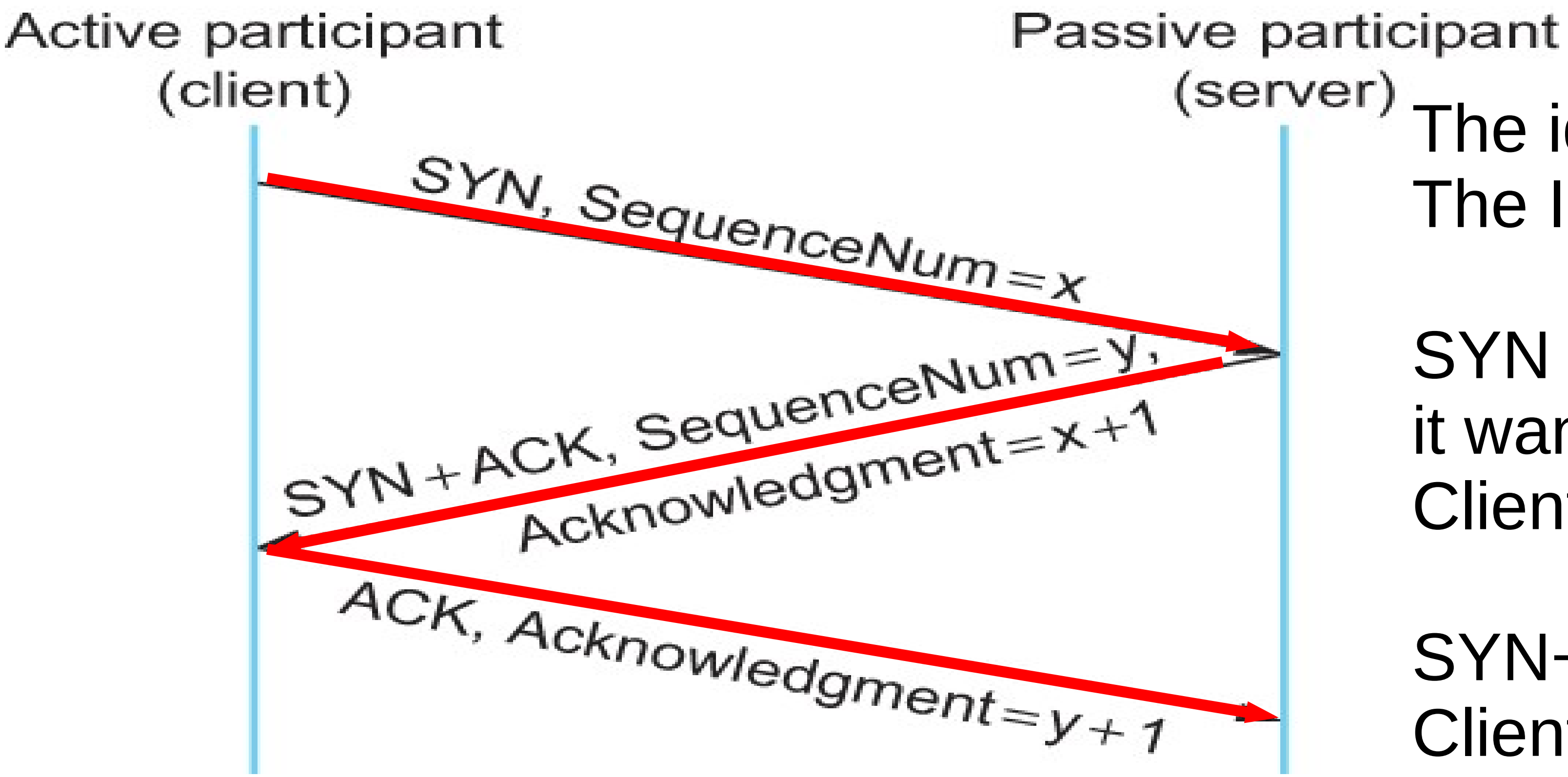
- TCP ISNs are clock based
  - 32 bits, increments in 4 microseconds
  - 4.55 hours wrap around time

TCP seq. numbers, ACKs



simple telnet scenario

TCP Three-way Handshake



Timeline for three-way handshake algorithm

The idea is to tell each other
The ISNs

SYN → Client tells server that it wants to open a connection, Client's ISN = x

SYN+ ACK → Server tells Client → Okay → Server's ISN = y, ACK = CLSeq + 1

Why increment by 1?

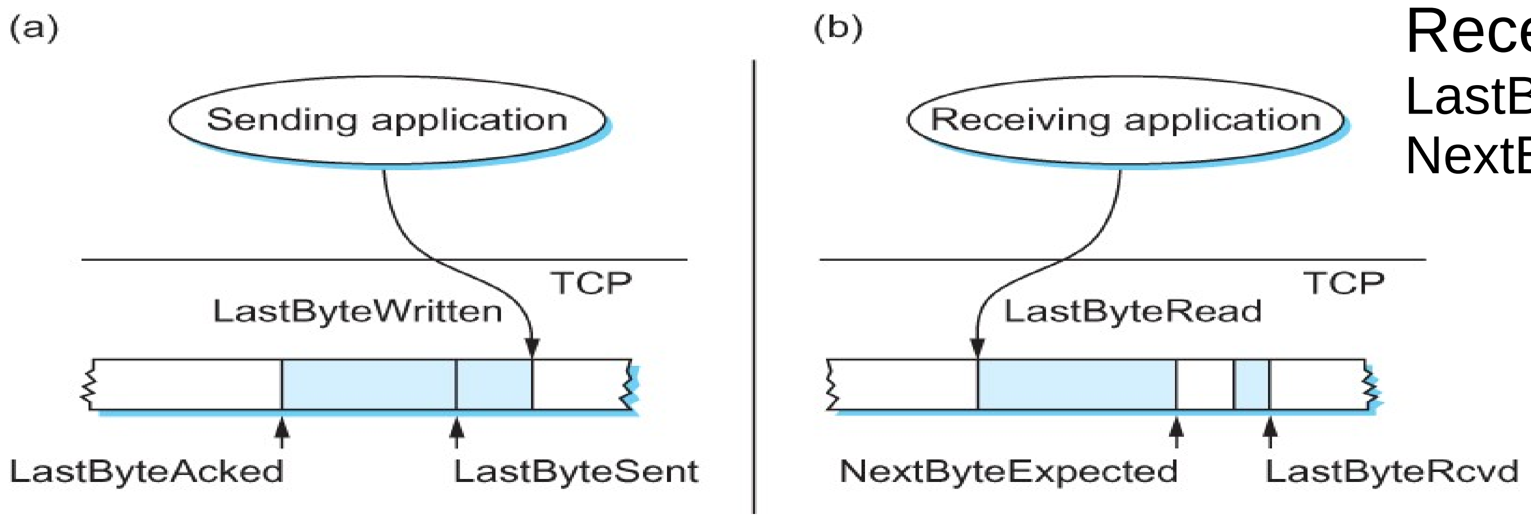
Sliding Window Revisited

Sending Side

LastByteAcked ≤ LastByteSent
LastByteSent ≤ LastByteWritten

Receiving Side

LastByteRead < NextByteExpected
NextByteExpected ≤ LastByteRcvd + 1



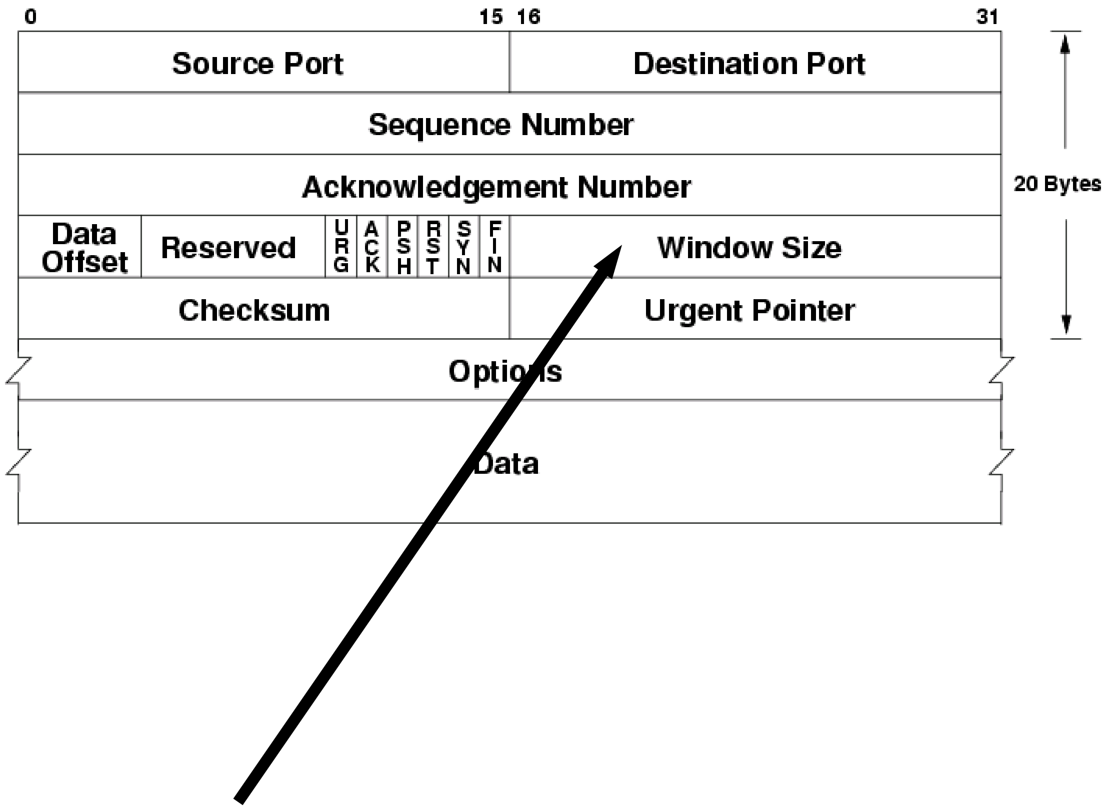
Relationship between TCP send buffer (a) and receive buffer (b).

TCP flow control

- receiver "advertises" free buffer space in the header

- sender limits amount of unacked ("in-flight") data to receiver's **`rwnd`** value

- guarantees receive buffer will not overflow

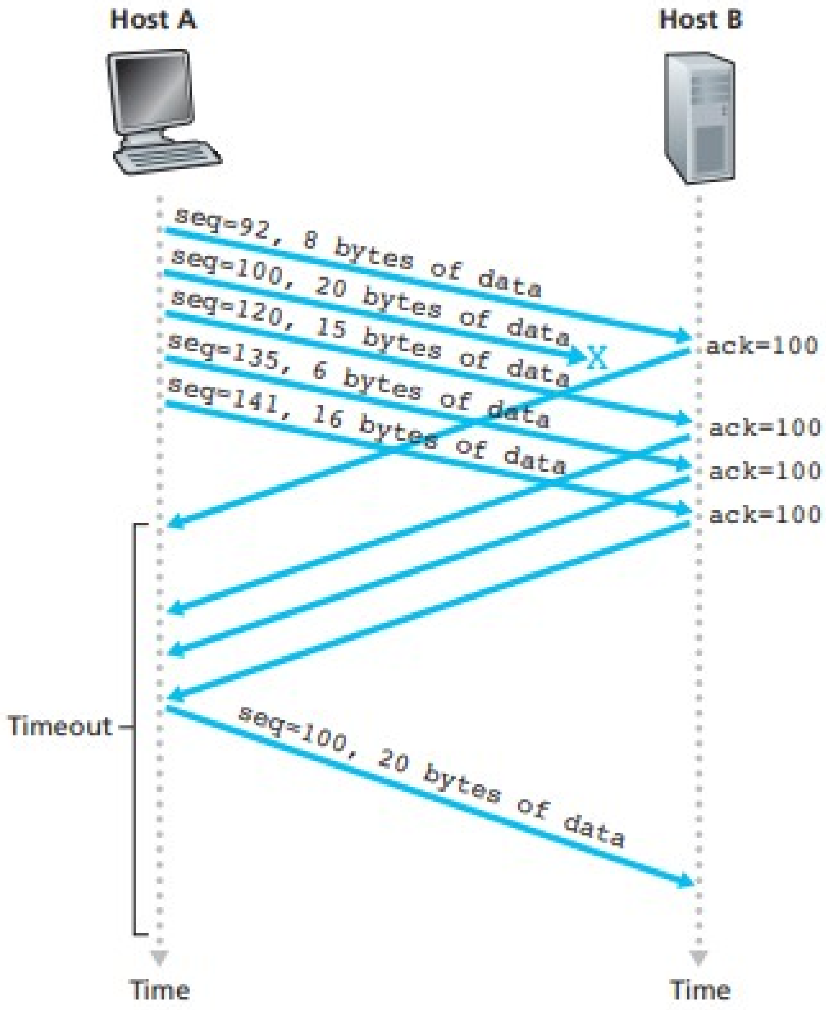


TCP Fast Retransmission

Timeouts are wasteful

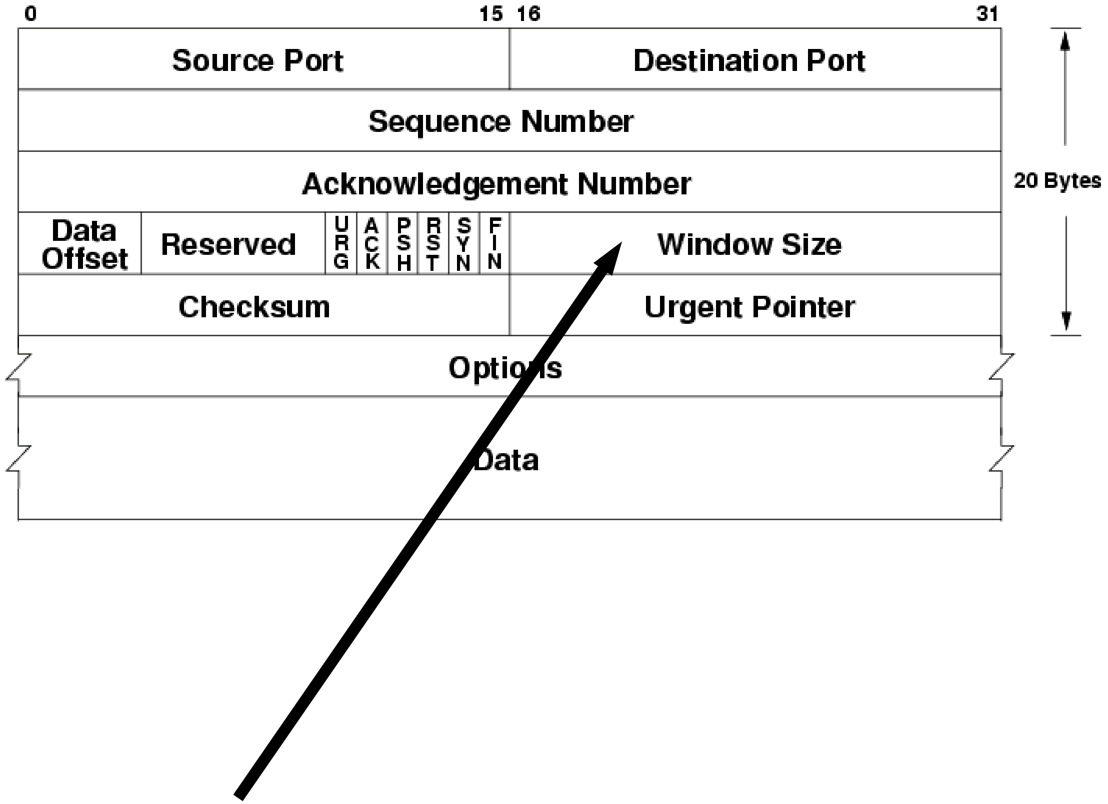
Triple duplicate ACKs

Retransmits before timeout



TCP flow control

- receiver "advertises" free buffer space in the header
- sender limits amount of unacked ("in-flight") data to receiver's **`rwnd`** value
- guarantees receive buffer will not overflow



Defining Fairness: Flows

“fair” to whom?   – Should be Fair to a Flow

What is a flow?

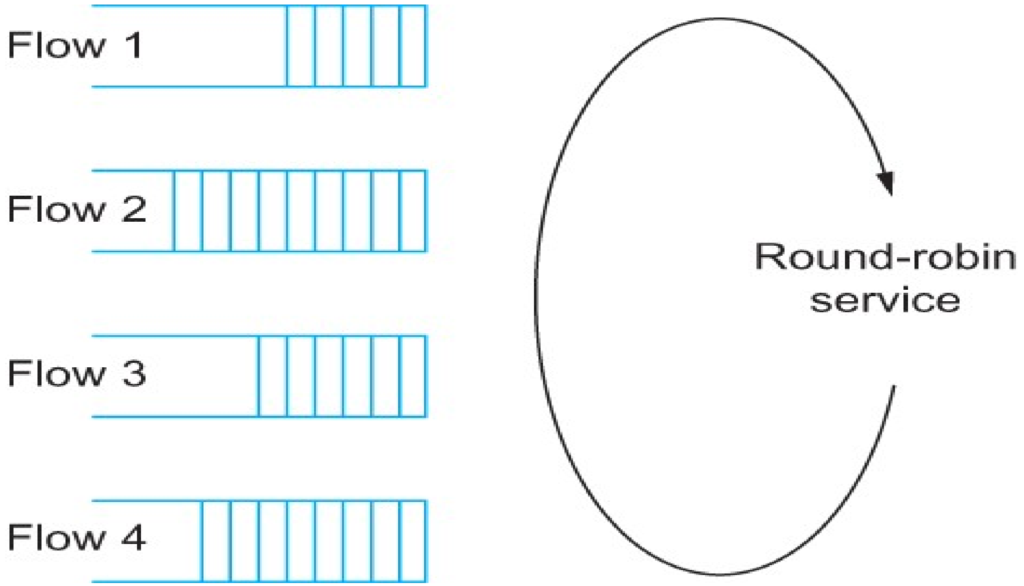
Combination of <Src IP, Src Port, Dst IP, Dst Port>

Fair Queuing

- Fair Queuing
  - FIFO does not discriminate between different traffic sources, or
  - it does not separate packets according to the flow to which they belong.
  - Fair queuing (FQ) maintains a separate queue for each flow

Queuing Disciplines

- Fair Queuing



Round-robin service of four flows at a router

Min Max Fair queuing

- Assume ***n*** clients
- Channel capacity **C**
- Give **c/n** to each client
  - If C1 does not want c/n
  - Divide the excess capacity equally among others
  - So everyone else gets c/n + (c/n – c1)/n-1
  - Repeat for C2 and others

Min Max Fair queuing Example

- Assume ***n*** clients - 5
- Channel capacity **C - 50**
- Give **c/n** to each client – **10/client**
  - If C1 does not want c/n – **4 extra**
  - Divide the excess capacity equally among others
  - Everyone else gets c/n + (c/n – c1)/n-1, **C1 → 5, C2..C5 - 11**
  - Repeat for C2 and others

Flow Control vs Congestion Control

- Flow Control:
  - Between end hosts
- Congestion Control:
  - In the network

Congestion Control

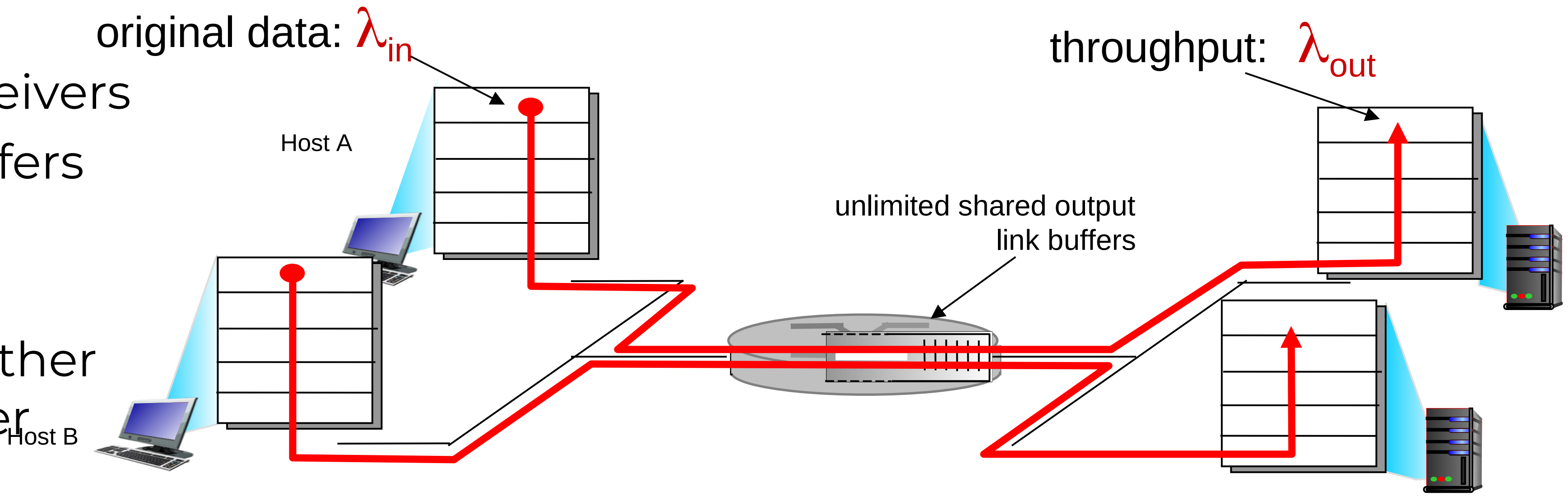
Principles of congestion control

congestion:

- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

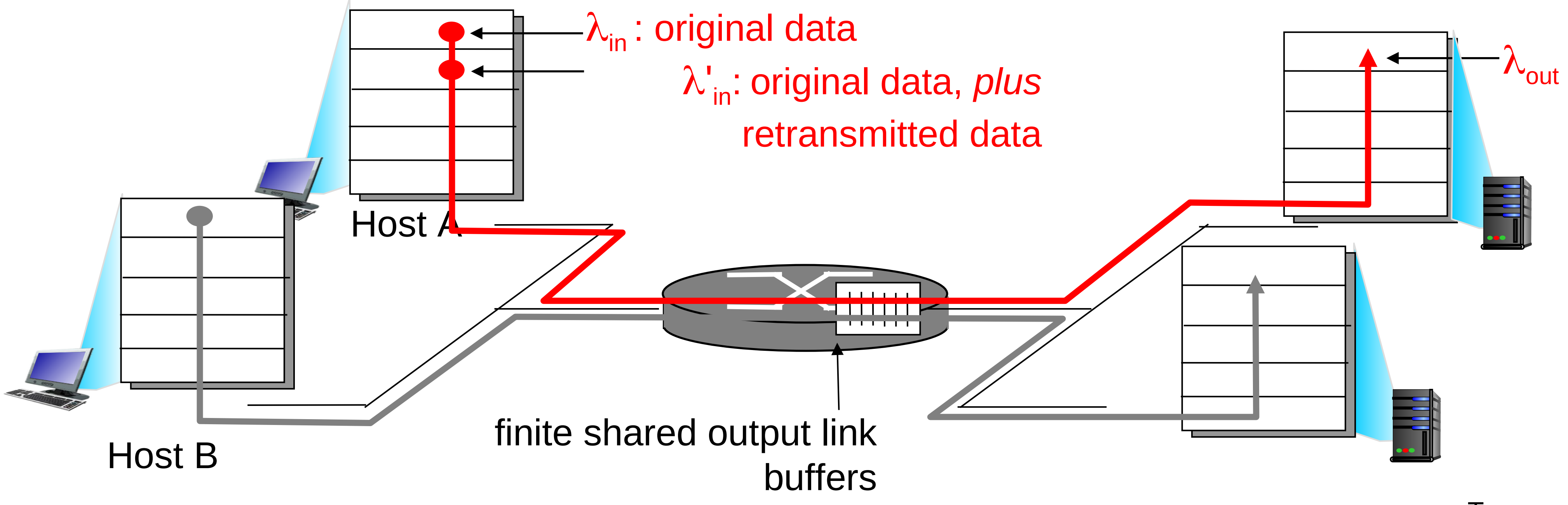
Congestion: scenario 1

- three senders, two receivers
- one router, infinite buffers
- output link capacity: R
- The router can only transmit one –... and either buffer or drop the other
- If many packets arrive,
- Buffer overflow



Causes/costs of congestion: scenario 2

- one router, *finite* buffers
- sender retransmission of timed-out packet
  - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$ $\geq$
  - transport-layer input includes *retransmissions* : λ_{in} λ_{in}



Metrics: Throughput vs Delay

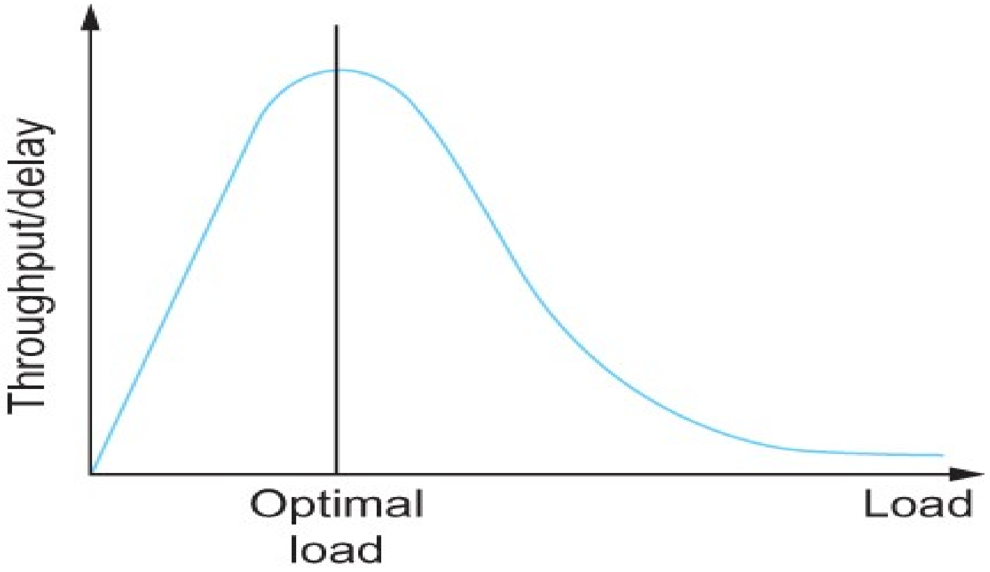
High throughput –

- Throughput: measured performance of a system –E.g., number of bits/second of data that get through
- Low delay –
- Delay: time required to deliver a packet or message –E.g., number of ms to deliver a packet •
- These two metrics are sometimes at odds –
  - More packets = more queuing

Issues in Resource Allocation

- Evaluation Criteria
  - Effective Resource Allocation

power of the network.
Power = Throughput/Delay



Ratio of throughput to delay as a function of load

Issues in Resource Allocation

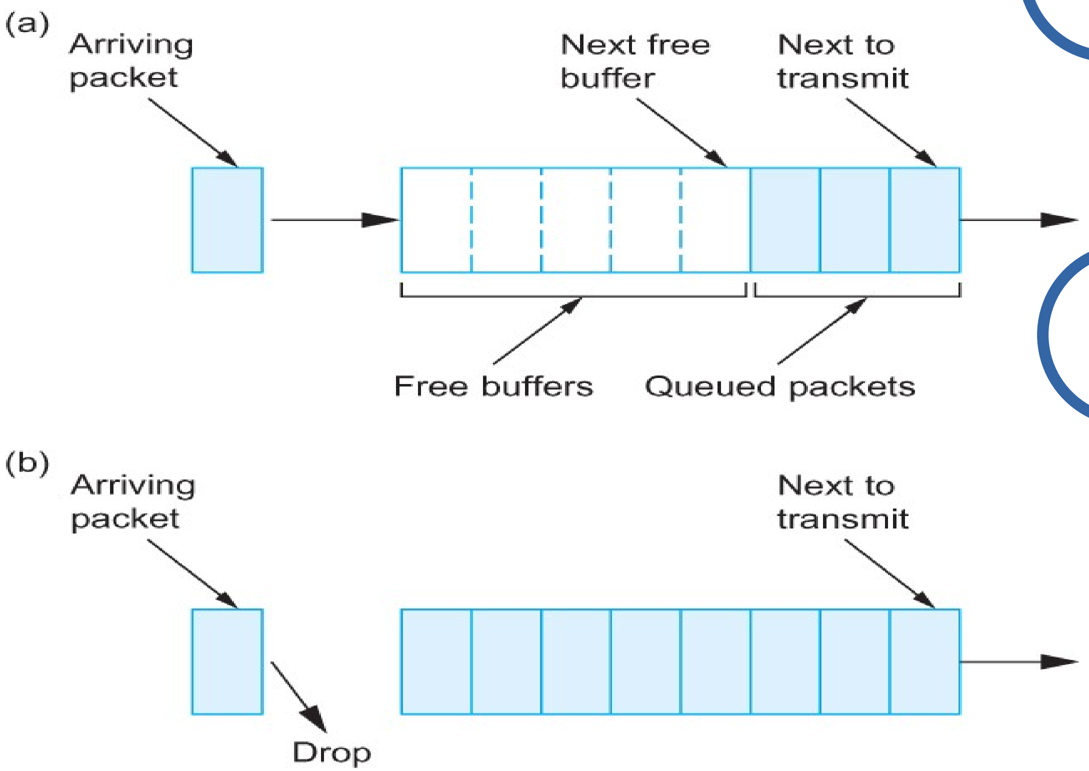
- Evaluation Criteria
  - Fair Resource Allocation
    - The effective utilization of network resources is not the only criterion for judging a resource allocation scheme.
    - We want to be “fair”
    - Equal share of bandwidth

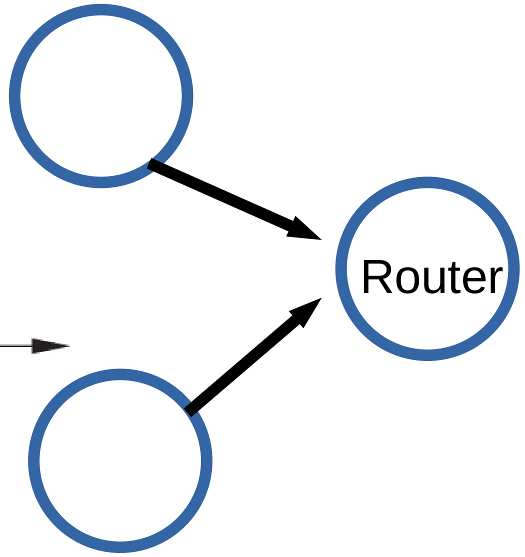
But, what if the flows traverse different paths?

Open problem, often determined by economics

Queuing Disciplines

Simplest – FIFO
and drop tail





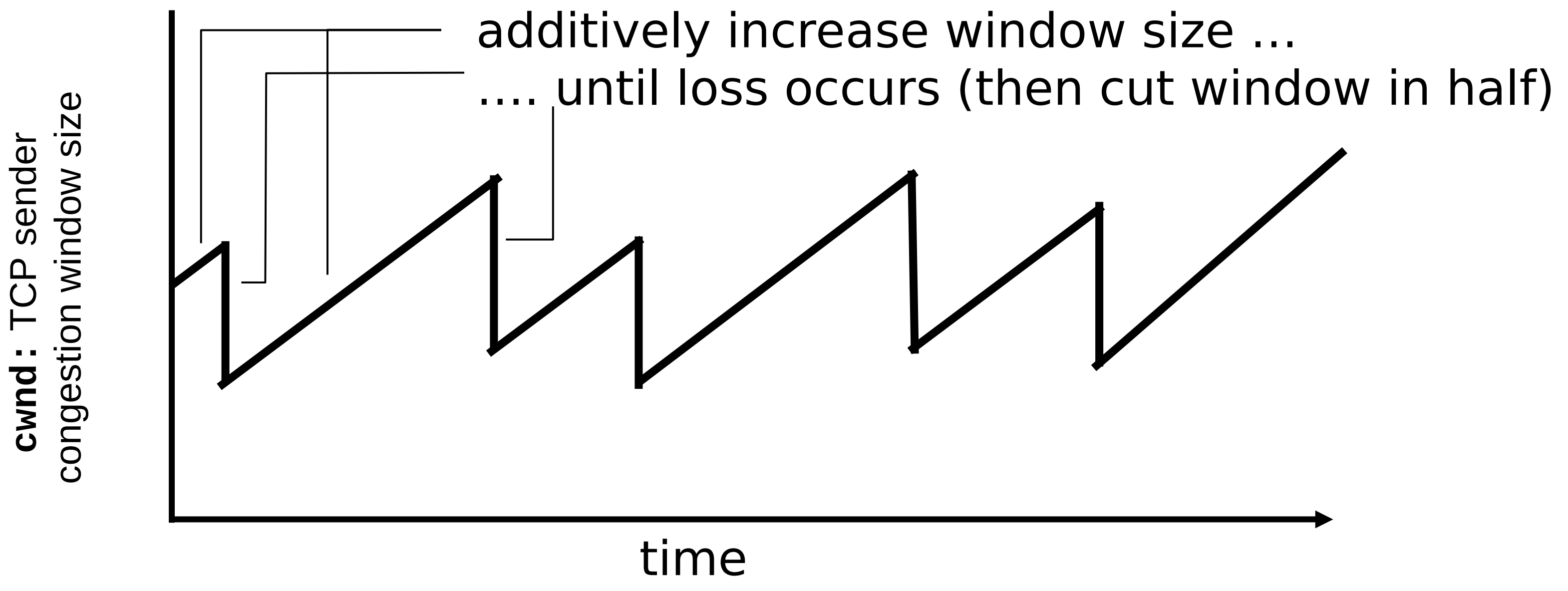
(a) FIFO queuing; (b) tail drop at a FIFO queue.

What are the problems?

TCP Congestion Control

What is the basic idea?

AIMD saw tooth
behavior: probing
for bandwidth



TCP Congestion Control

- Each source determines available capacity
- Max many packets is allowed to have in transit - window
- Congestion window = # of unacked bytes
- MaxSendWindow = min(congestion window, receiver window)

- How do you change congestion window?
  - Decrease on losing a packet (back off)
  - Increase on successful send

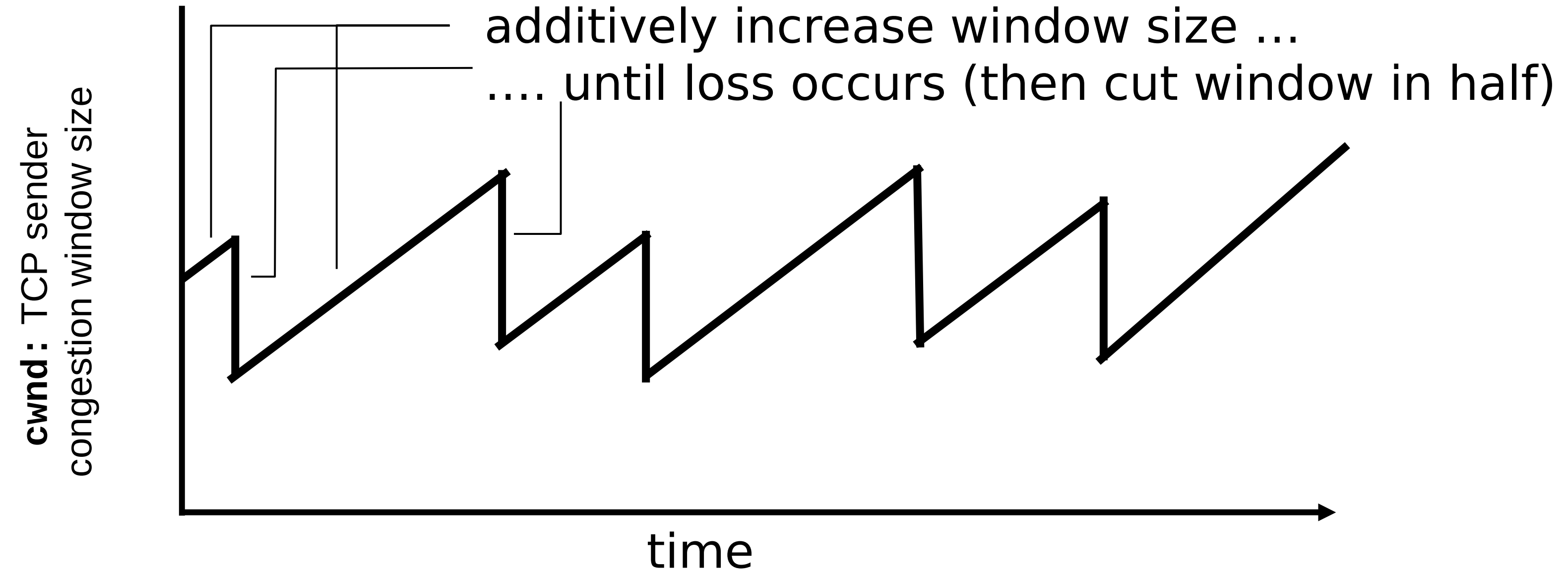
How much to increase and decrease?

- Additive Increase, Multiplicative Decrease (AIMD)

How much to increase and decrease?

- *approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **cwnd** by 1 MSS every RTT until loss detected
  - *multiplicative decrease*: cut **cwnd** in half after loss

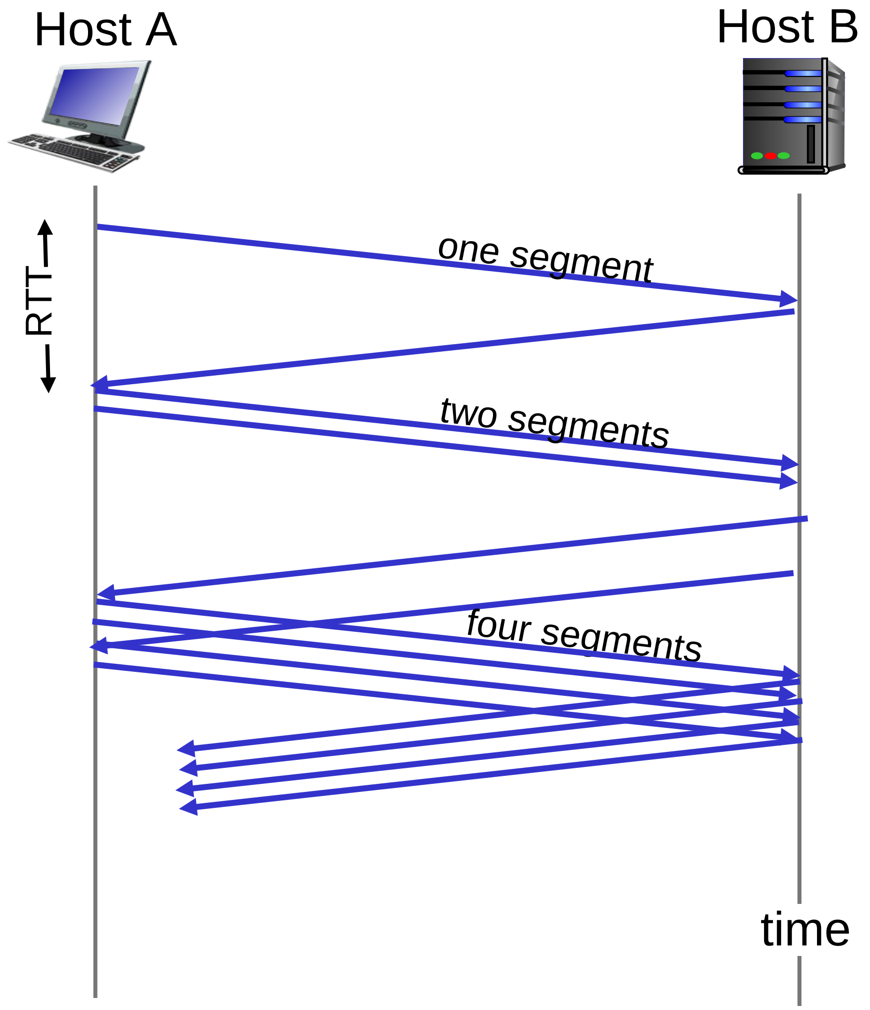
AIMD saw tooth behavior: probing for bandwidth



TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - double **cwnd** every RTT
  - done by incrementing **cwnd** for every ACK received

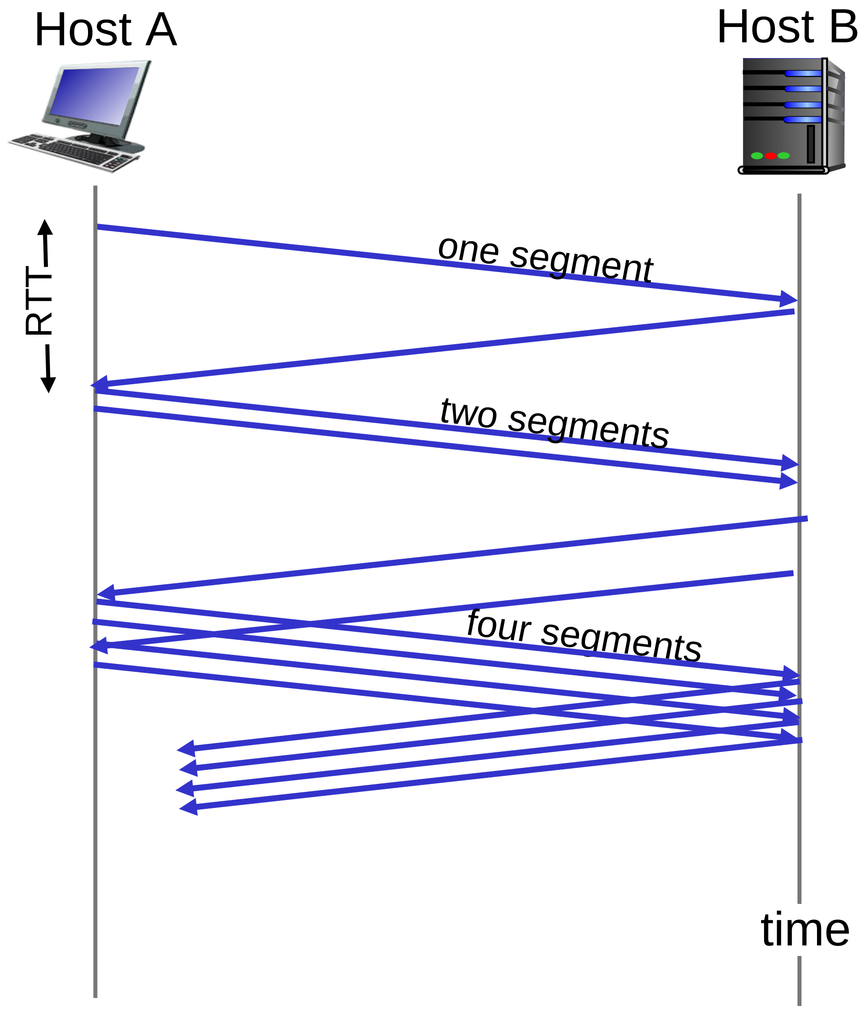
- *summary:* initial rate is slow but ramps up exponentially fast



TCP Slow Start

Why not start with a large window?

Why not increase one by one?



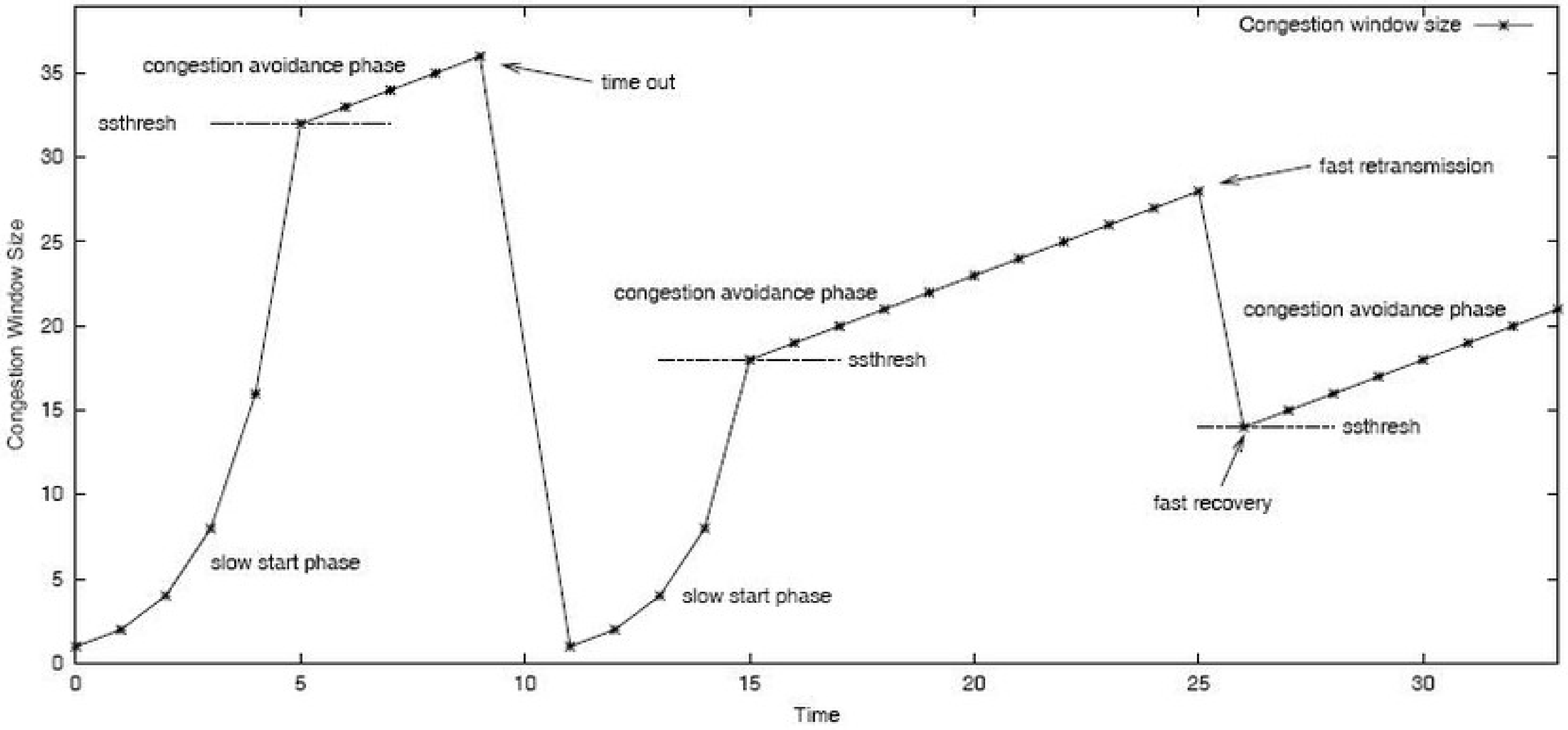
TCP: detecting, reacting to loss

- loss indicated by timeout:
  - **cwnd** set to 1 MSS;
  - window then grows exponentially (as in slow start) to threshold, then grows linearly
- loss indicated by 3 duplicate ACKs: TCP RENO
  - dup ACKs indicate network capable of delivering some segments
  - **cwnd** is cut in half window then grows linearly
- TCP Tahoe always sets **cwnd** to 1 (timeout or 3 duplicate acks)

TCP:Two types of loss

- Triple duplicate ack
  - Do a multiplicative decrease, keep going
- Timeout
  - Reset CWND to 1
  - Take advantage of

TCP Slow Start and congestion avoidance



https://www.researchgate.net/figure/3-TCP-slow-start-phase-and-congestion-avoidance-phase_fig3_225731524

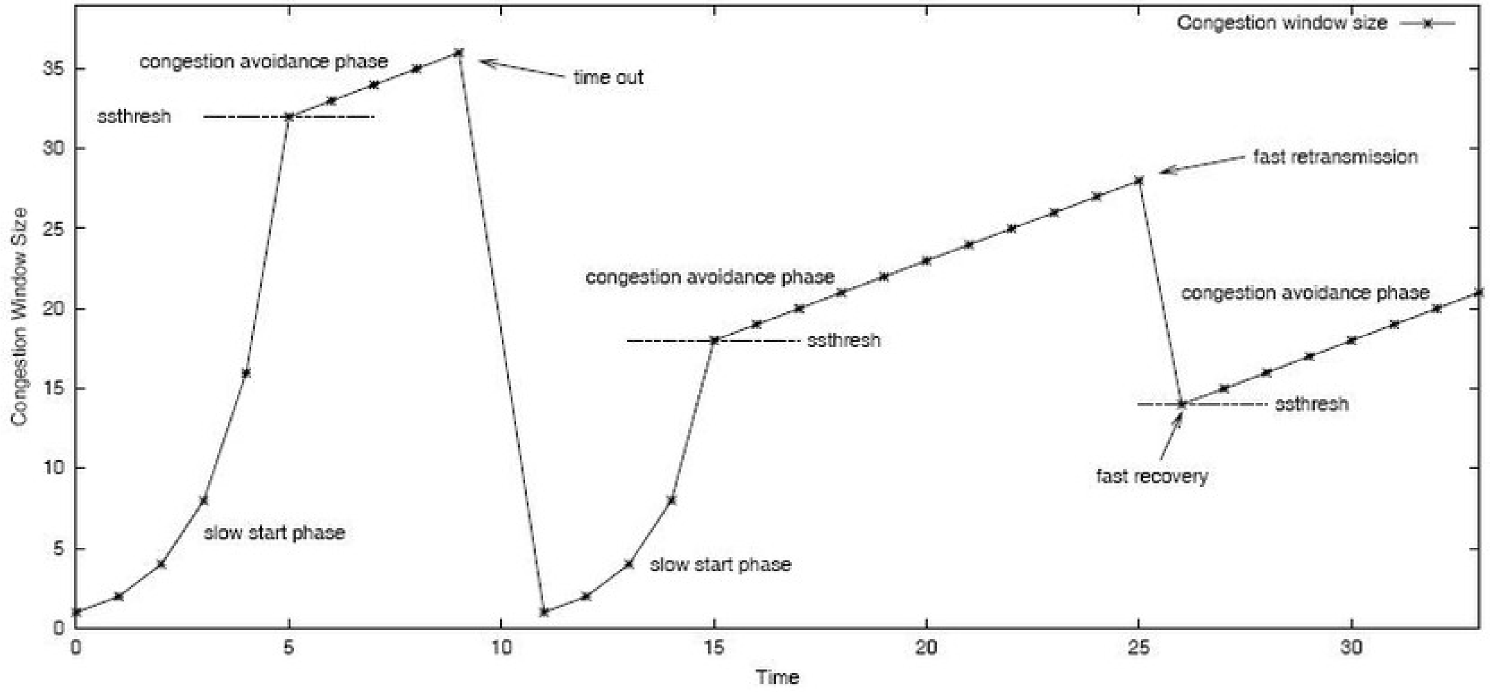
How to set ssthresh?

Initially – Randomly high

Later – adjusted as congestion happens

TCP Congestion Summary

CWND < Threshold → Slow Start, Exponential increase
CWND > Threshold → Congestion Avoidance, Linear increase
Triple Duplicate ACK → Threshold = CWND/2, CWND = CWND/2
Timeout → Threshold = CWND/2, CWDN = 1 (or 3)



TCP Throughput

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

Ignore slow start, assume long TCP flow

Let W be the window size

Throughput = W/RTT

After loss, throughput = W/2*RTT

Average throughput = 0.75W/RTT

Problems with Fast Links

Consider the high speed link:

9000 byte segments

100ms RTT

100Gbps/second throughput

Throughput = 0.75W/RTT

So, WindowSize (w) = Throughput * RTT / 0.75

W = 1,481,481,444 segments

Problems with Fast Links

TCP assumes all losses are due to congestion

Throughput = (1.22*MSS)*(RTT/sqrt(Loss))

What is the loss rate to maximize 100Gbps pipe with
9000 bytes segments and 100ms RTT? Hint – must be very very low

https://www.switch.ch/network/tools/tcp_throughput/

TCP Throughput

TCP average throughput as a function of window size and RTT?
Ignore slow start, assume long TCP flow

Let W be the window size

Throughput = W/RTT
After loss, throughput = W/2*RTT
Average throughput = 0.75W/RTT

Throughput = (1.22*MSS)*(RTT/sqrt(Loss)) ← Magic formula

What is the loss rate to maximize 100Gbps pipe with 9000 bytes segments and 100ms RTT? Hint – must be very very low

Reading

https://book.systemsapproach.org/congestion/tcpcc.html#tcp-congestion-control

TCP Throughput

TCP average throughput as a function of window size and RTT?
Ignore slow start, assume long TCP flow

Let W be the window size

Throughput = W/RTT
After loss, throughput = W/2*RTT
Average throughput = 0.75W/RTT

Throughput = (1.22*MSS)*(RTT/sqrt(Loss)) ← Magic formula

What is the loss rate to maximize 100Gbps pipe with 9000 bytes segments and 100ms RTT? Hint – must be very very low