EE 542
Lecture 4: More Networking
Internet and Cloud Computing

Young Cho
Department of Electrical Engineering
University of Southern California

Laboratories

- Lab 1: Virtual Machines and Network
- Lab 2: AWS Cloud Hands-on Lab
- Lab 3: Network Measurements
- Lab 4: Fast Reliable File Transfer (L7)
- Lab 5: Fast Reliable File Transfer (Kernel)
- Lab 6: Phone-based IoT and Cloud
- Lab 7: Industrial IoT – xDot
- Lab 8: IIoT to Cloud
- Lab 9: Hadoop
- Lab 10: Message Passing Protocol
- Lab 11: Sparking Programming
- Lab 12: Machine Learning/Deep Learning

Upper Layers Networking

- Transport Layer
  - The delivery of data to the application
  - Reliable delivery
  - Error detection
  - Flow control
  - Congestion avoidance
  - TCP - Transmission Control Protocol
  - UDP - User Datagram Protocol
- Session Layer
  - Opening, closing and managing a session
  - NetBIOS - Network Basic Input Output System
  - PPTP - Point-to-Point Tunneling Protocol
  - RPC - Remote Procedure Call Protocol
  - SCP - Secured Copy Protocol
  - Often pushed back to application
- Presentation Layer
  - The delivery and formatting of information
  - Extended Binary Coded Decimal Interchange Code text file to an ASCII conversion
  - Encryption
  - Compression
  - Often pushed back to application
- Application Layer
  - Custom applications
  - FTP - File Tranfer Protocol
  - HTTP - HyperText Transfer Protocol
  - SMTP - Simple Mail Transfer Protocol
  - POP3 - Post Office Protocol version 3
  - TELNET - Terminal Emulation Protocol of TCP/IP
  - SSH - Secure Shell
  - Your own protocols

Transport vs. Network layer

- *Network layer:* logical communication between hosts
- *Transport layer:* logical communication between processes
  - relies on, enhances, network layer services

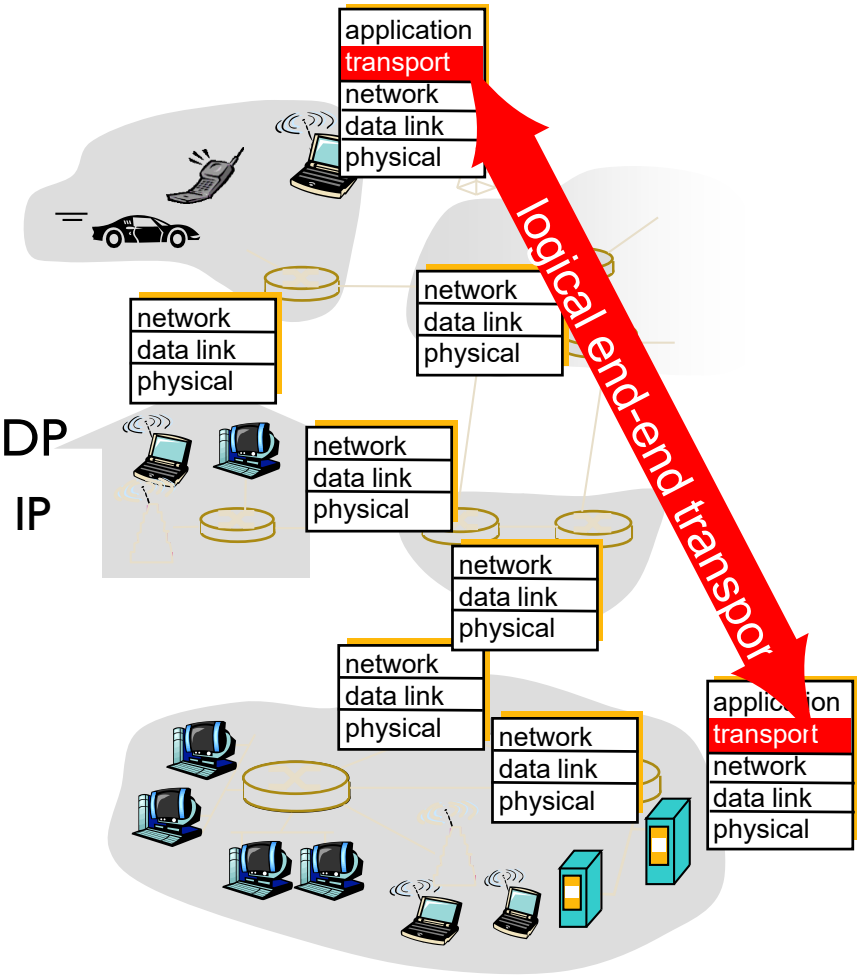
Postal service analogy:

kids sending letters to other kids

- processes = kids
- app messages = letters in envelopes
- hosts = houses
- transport protocol = two specific kids
- network-layer protocol = postal service

Internet transport-layer protocols

- Reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- Unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- Services not available:
  - delay guarantees
  - bandwidth guarantees



Multiplexing/Demultiplexing

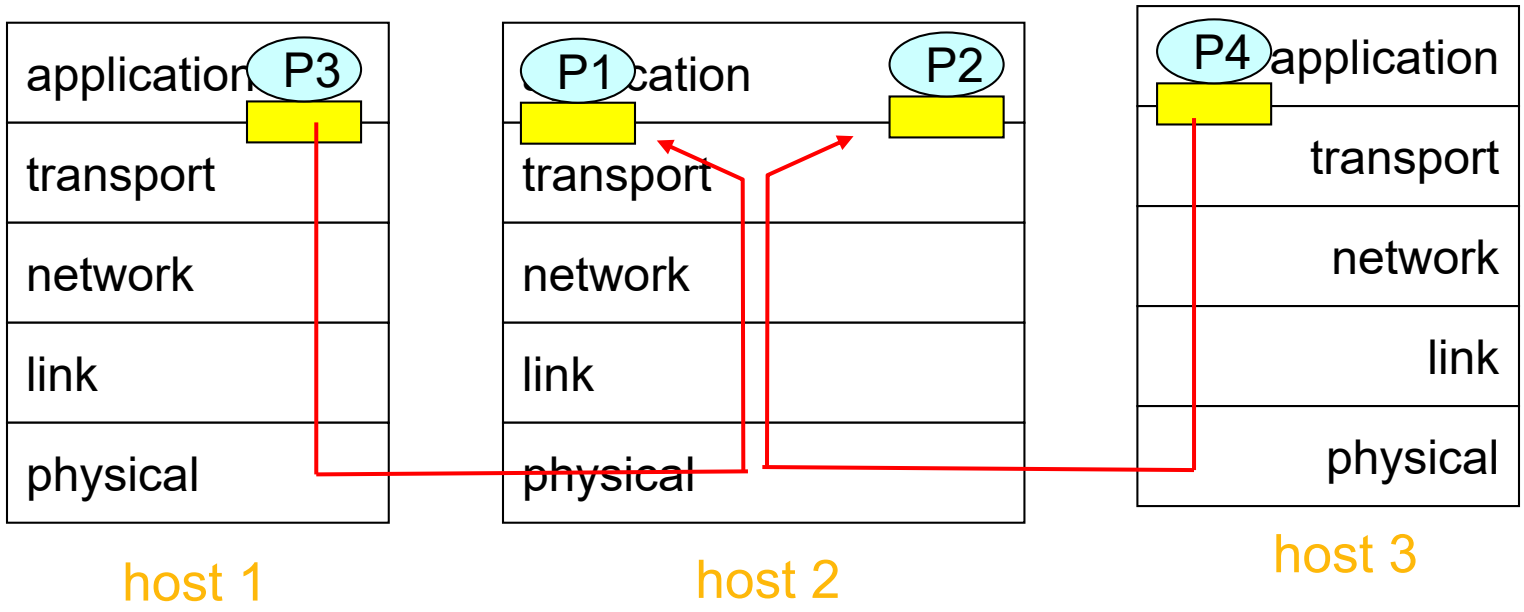
Demultiplexing at rcv host:

delivering received segments to correct socket

Multiplexing at send host:

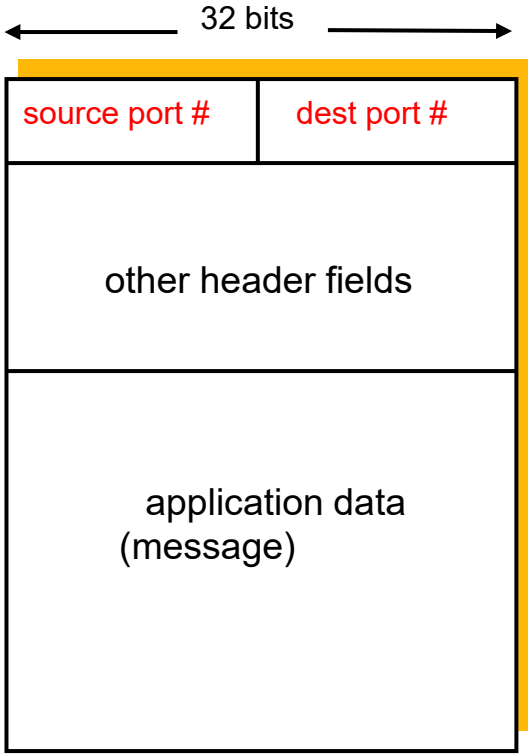
gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

= socket = process



Demultiplexing

- Host receives IP datagrams
  - source IP address, destination IP address
  - datagram carries 1 transport-layer segment
  - source, destination port number
  - host uses IP addresses & port numbers to direct segment to appropriate socket
- Connectionless demultiplexing
  - SP/DP forms 2-tuple Socket
  - SP provides return address
  - Used with UDP
- Connection-oriented demultiplexing
  - SIP/DIP/SP/DP forms 4-tuple Socket
  - SIP/SP provides return address



TCP/UDP segment format

UDP: User Datagram Protocol [RFC 768]

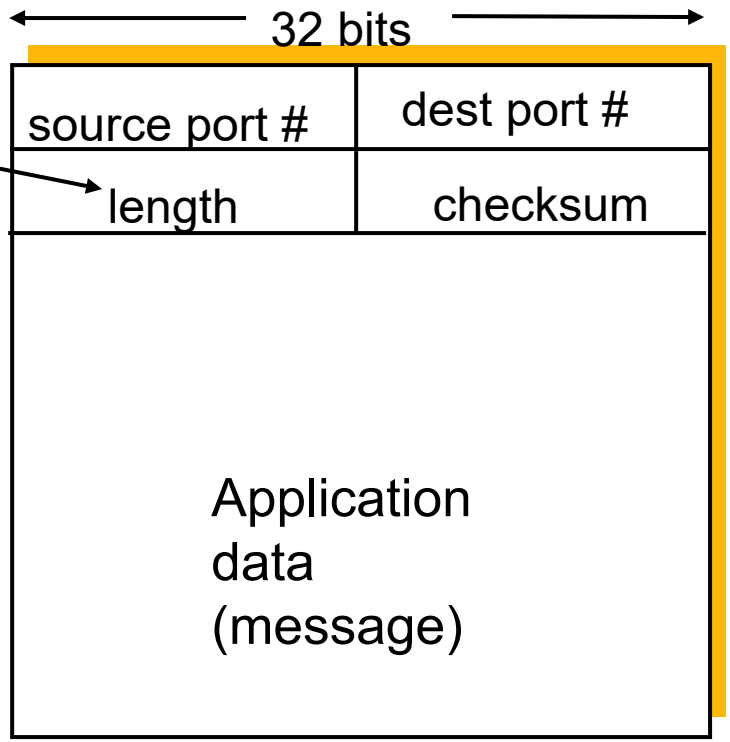
- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

UDP: more

- often used for streaming multimedia apps
  - loss tolerant
  - rate sensitive
- other UDP uses
  - DNS
  - SNMP
- reliable transfer over UDP: add reliability at application layer
  - application-specific error recovery!

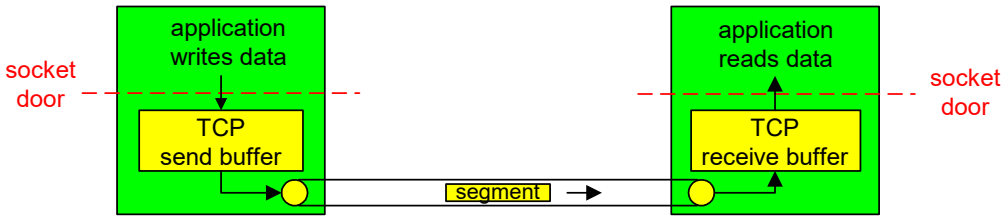


UDP segment format

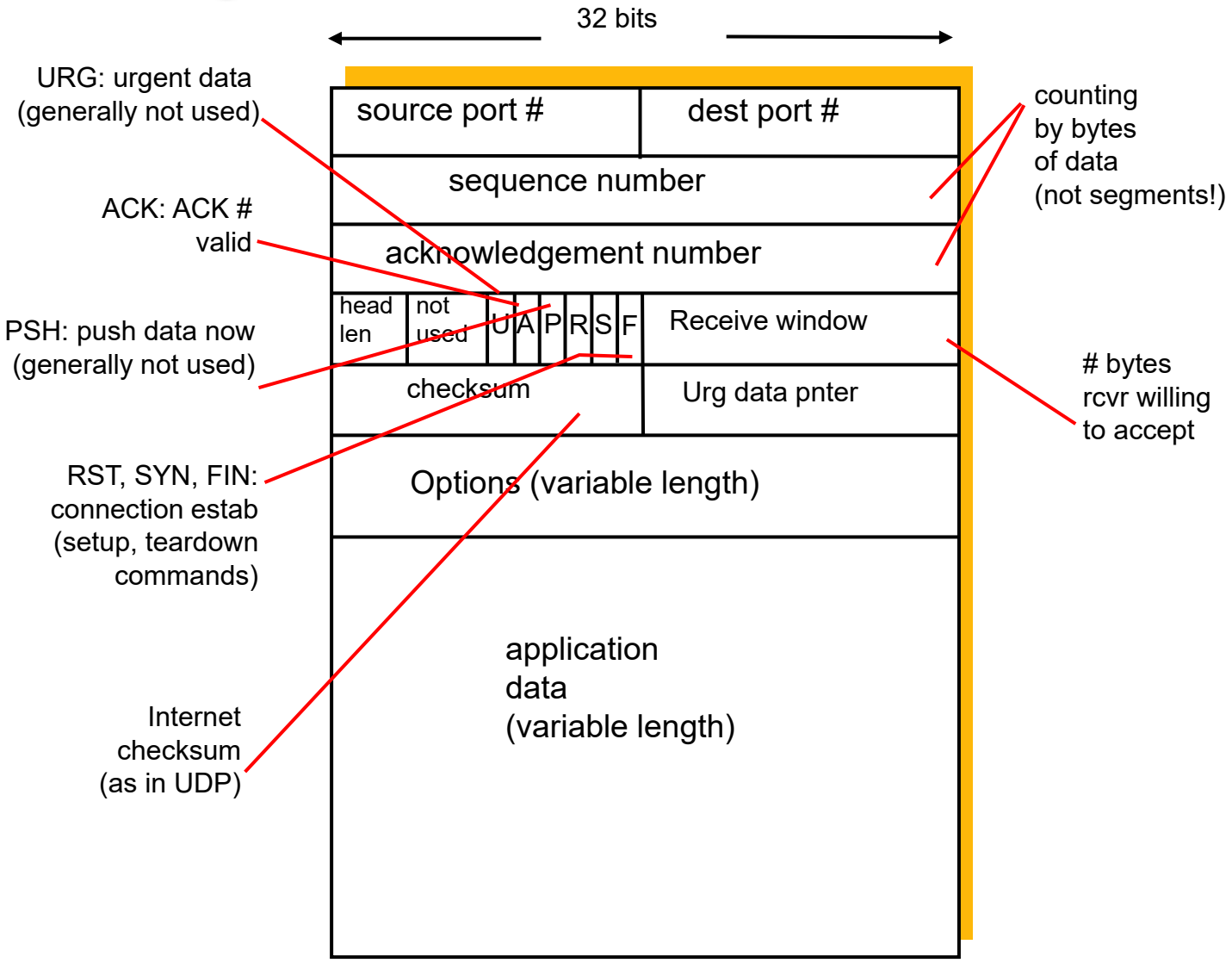
TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver
- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



TCP segment structure



TCP seq. #'s and ACKs

Seq. #'s:

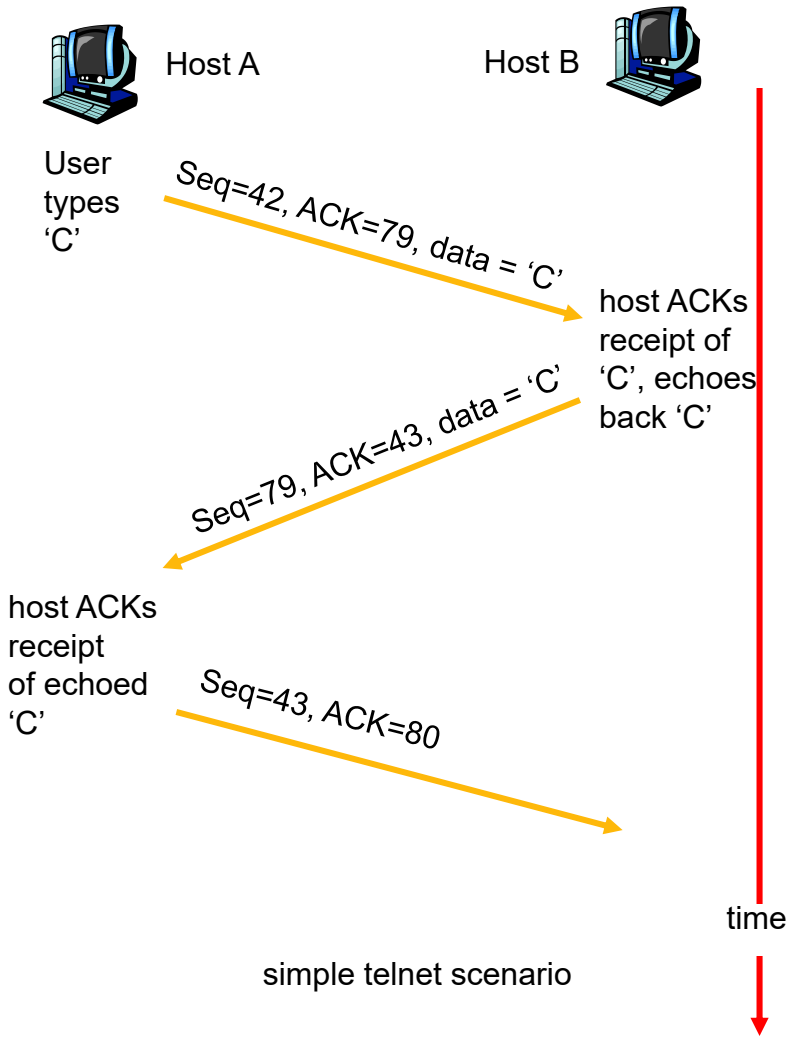
- byte stream "number" of first byte in segment's data

ACKs:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor

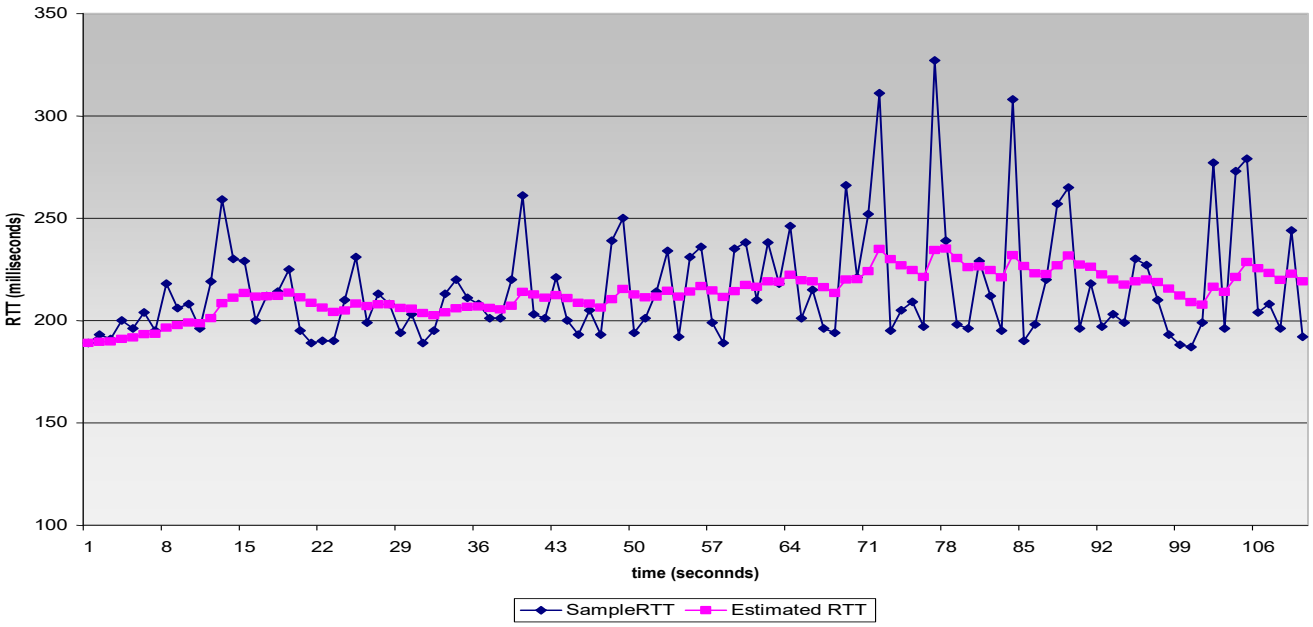


simple telnet scenario

TCP Round Trip Time

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

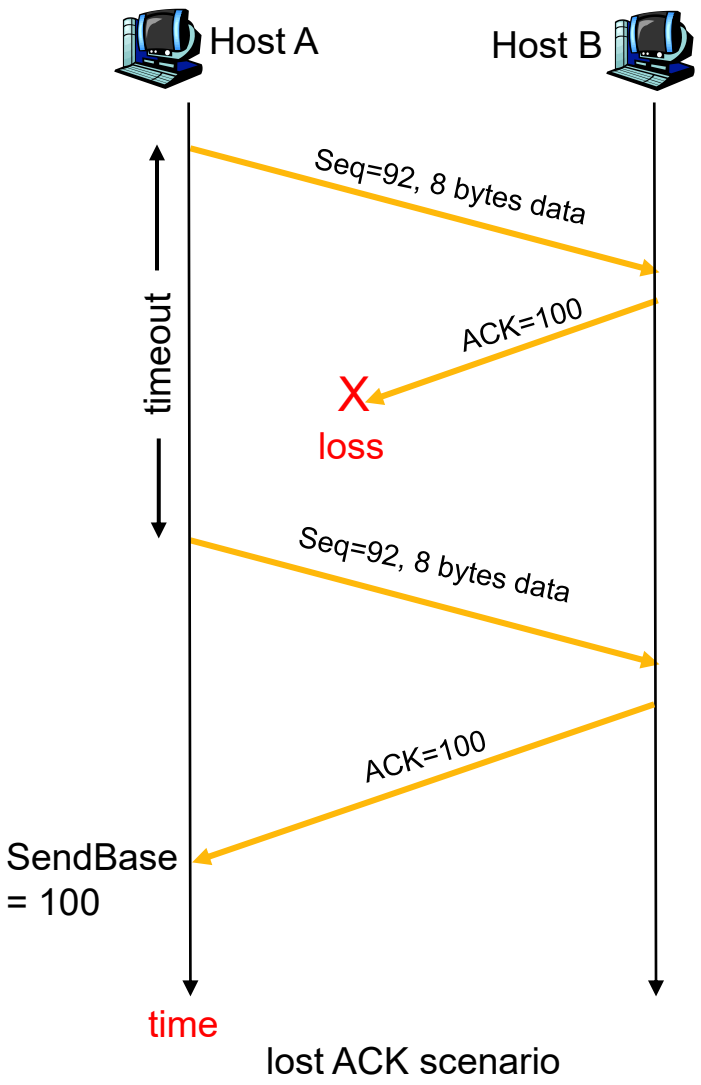
- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: α = 0.125



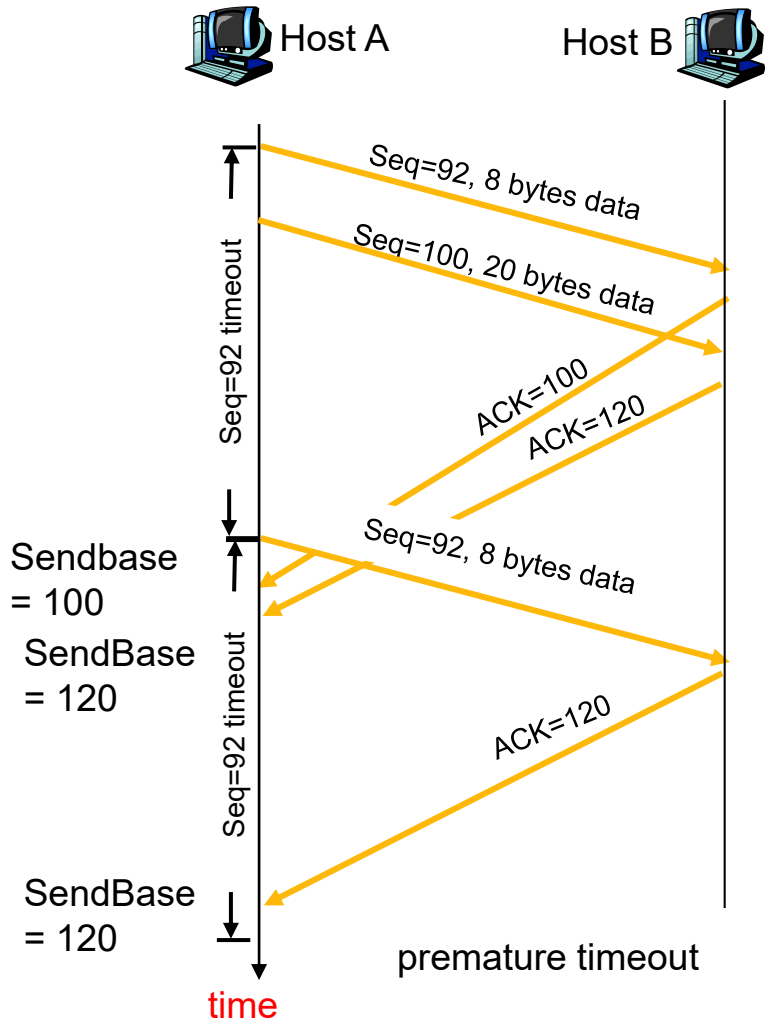
TCP reliable data transfer

- On top of IP's unreliable Service
- Pipelined segments
- Cumulative Acks
- Key to Reliable Delivery: Retransmission
  - TCP uses single retransmission timer
  - timeout events
  - duplicate acks

TCP: retransmission scenarios

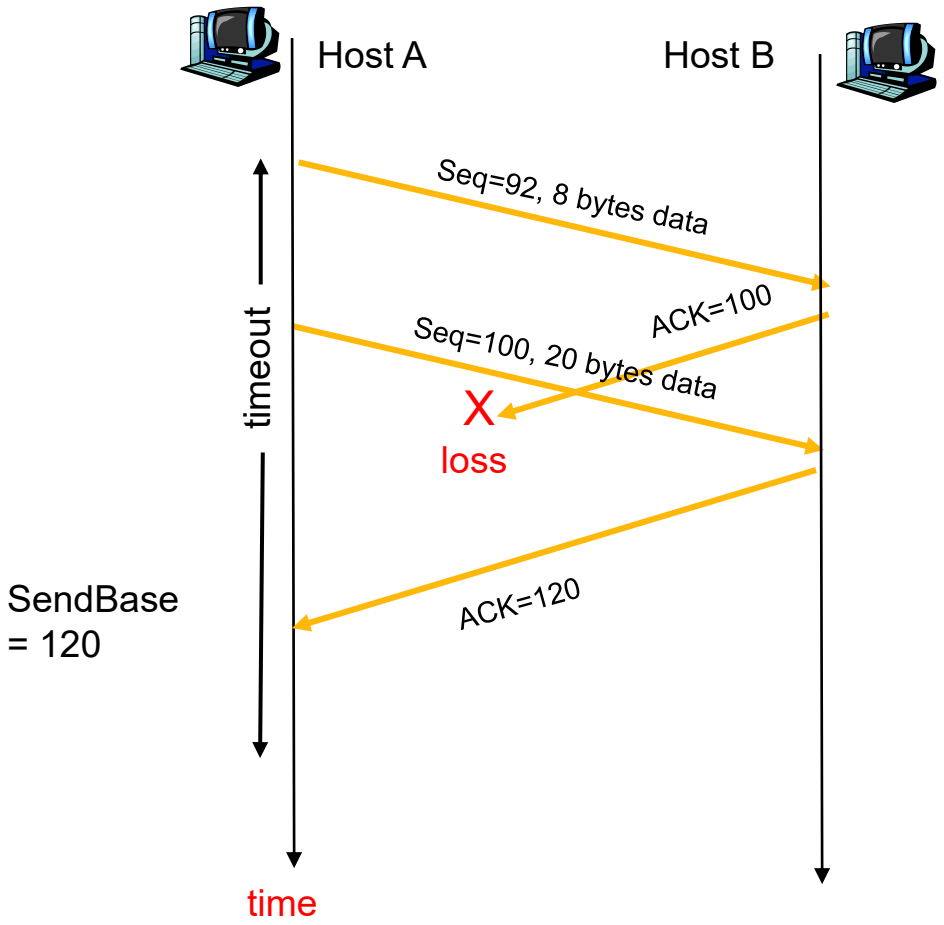


lost ACK scenario



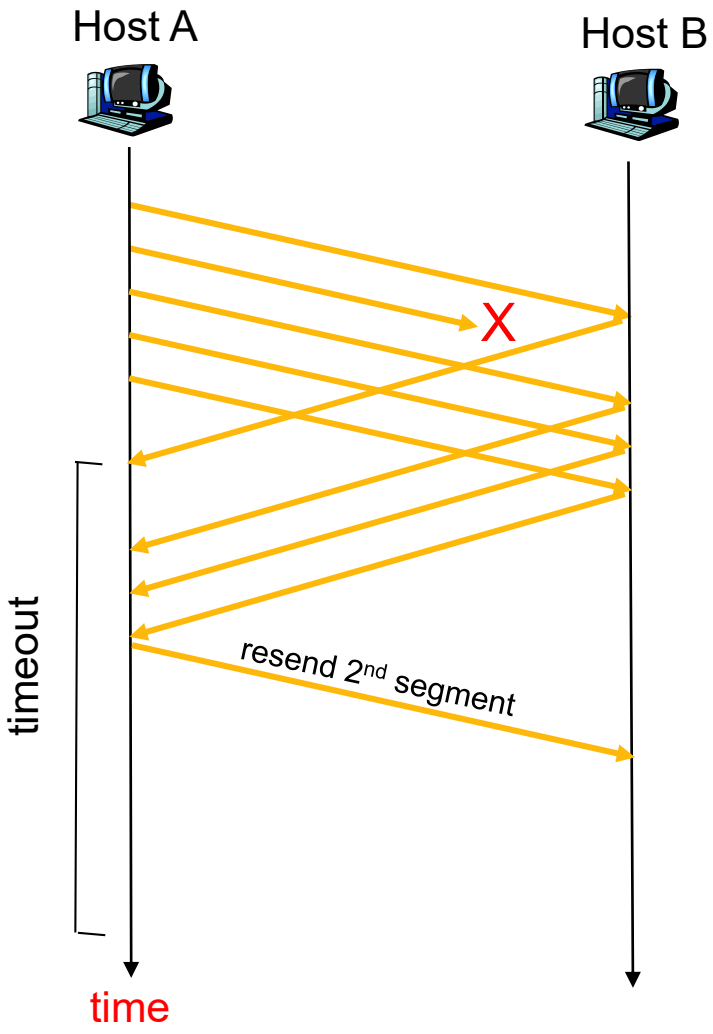
premature timeout

TCP retransmission scenarios



Cumulative ACK scenario

Fast Retransmit

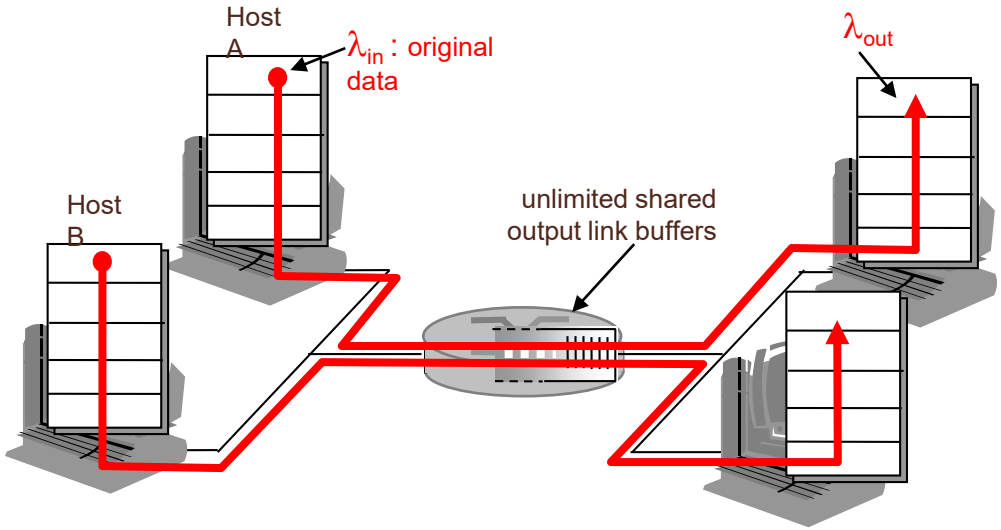


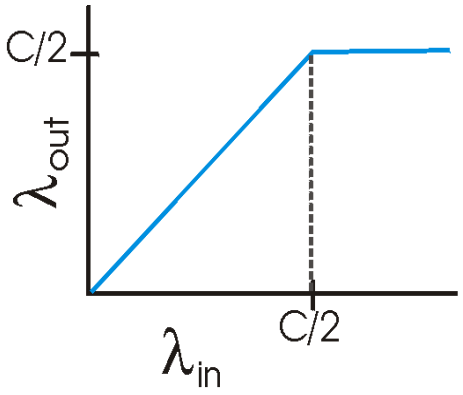
Congestion

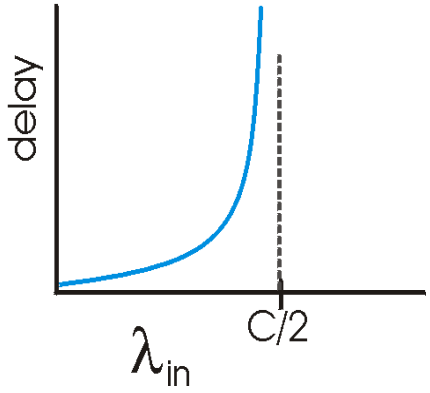
- Sending too much data too fast
  - different from flow control
- Lost packets
  - buffer overflow at routers
- Long delays
  - queueing in router buffers

Causes/costs of congestion: scenario 1

- two senders, two receivers
- one router, infinite buffers
- no retransmission



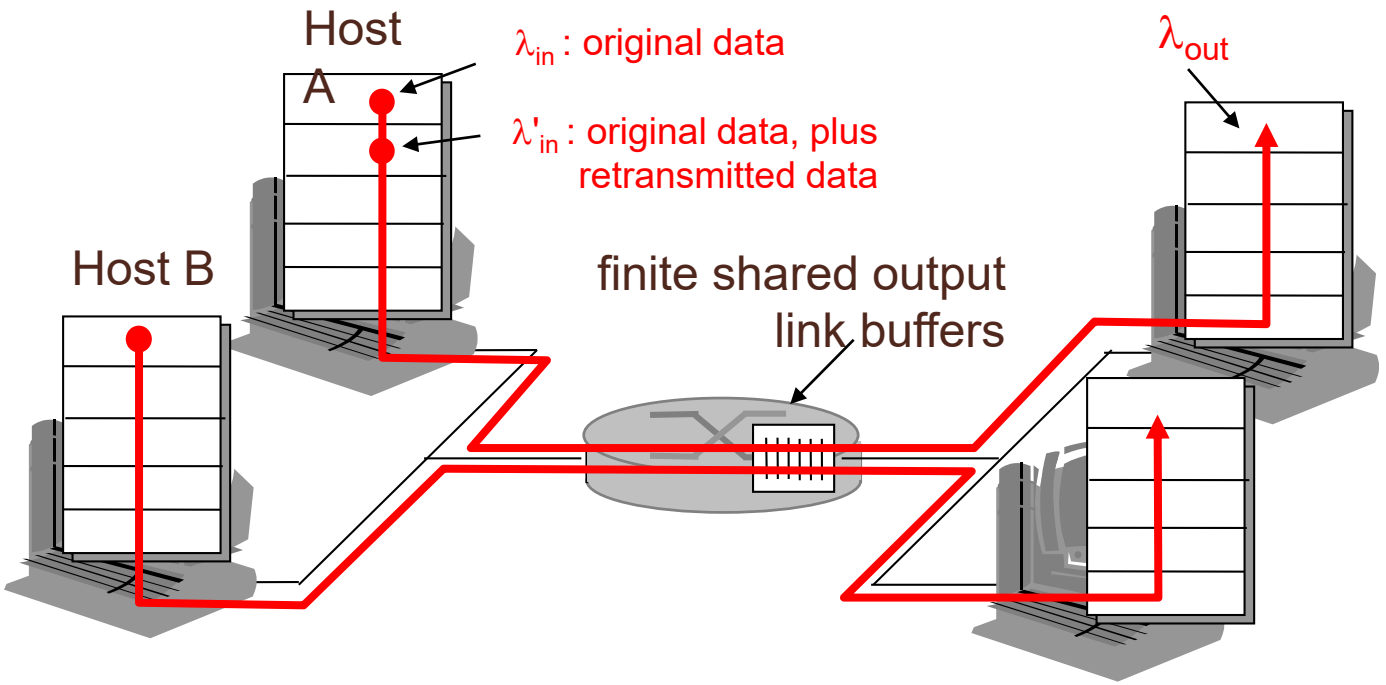




- large delays when congested
- maximum achievable throughput

Causes/costs of congestion: scenario 2

- one router, *finite* buffers
- sender retransmission of lost packet



Two broad approaches towards congestion control

End-end congestion control:

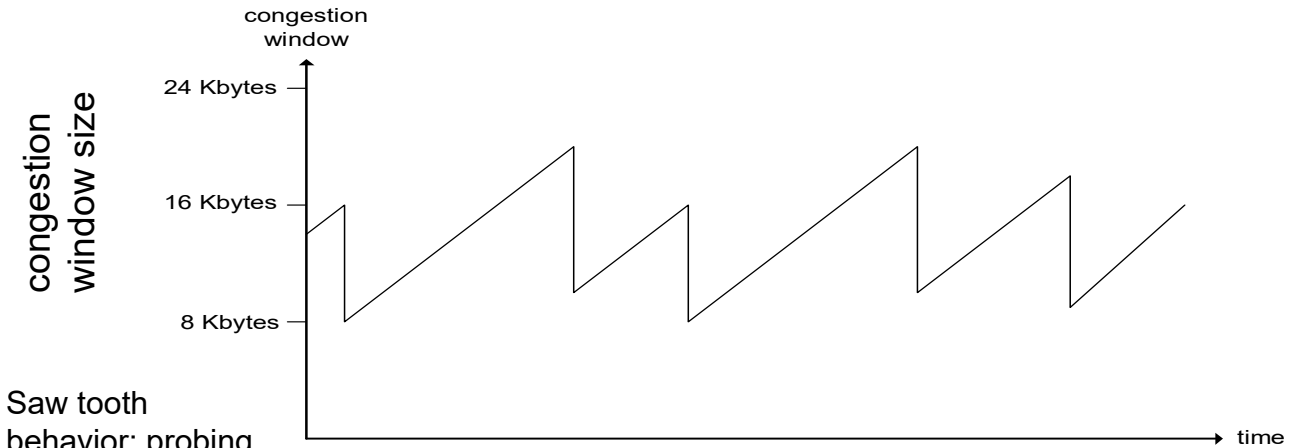
- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- Approach taken by TCP

Network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate sender should send at

TCP congestion control

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



Saw tooth behavior: probing for bandwidth

TCP Congestion Control

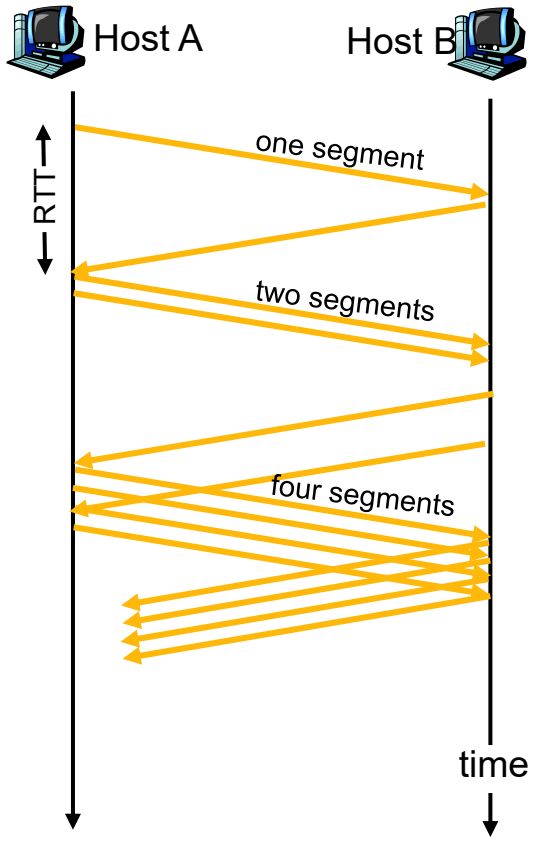
- Sender limits transmission
- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- **`CongWin`** is dynamic
  - function of perceived network congestion
- Three mechanisms
  - Additive Increase Multiplicative Decrease
  - Slow start
  - Conservative after timeout events

TCP Slow Start

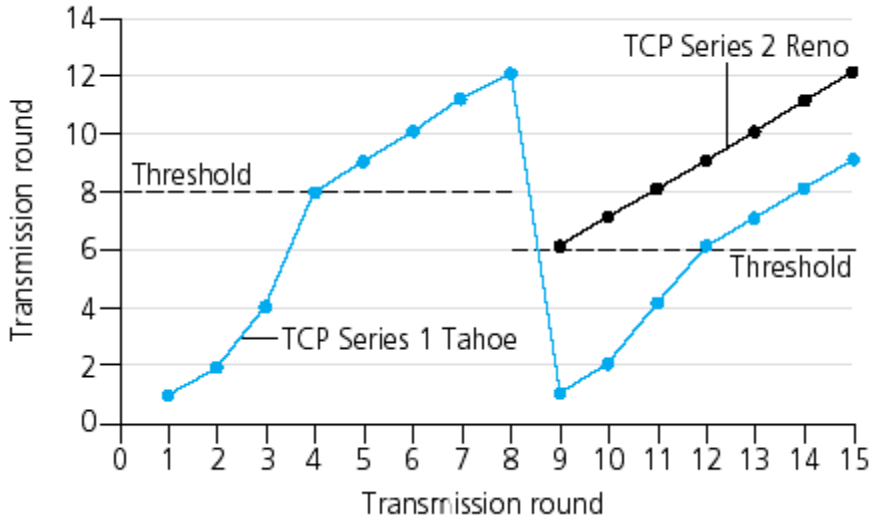
- When connection begins
  - `CongWin` = 1 MSS (Maximum Segment Size)
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- Available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate
- When connection begins
  - increase rate exponentially fast until first loss
  - increase rate exponentially until first loss event:
  - double CongWin every RTT
  - done by incrementing CongWin for every ACK received



Refinement

When should the exponential increase switch to linear?

- **`CongWin`** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, set Threshold to 1/2 of CongWin before loss

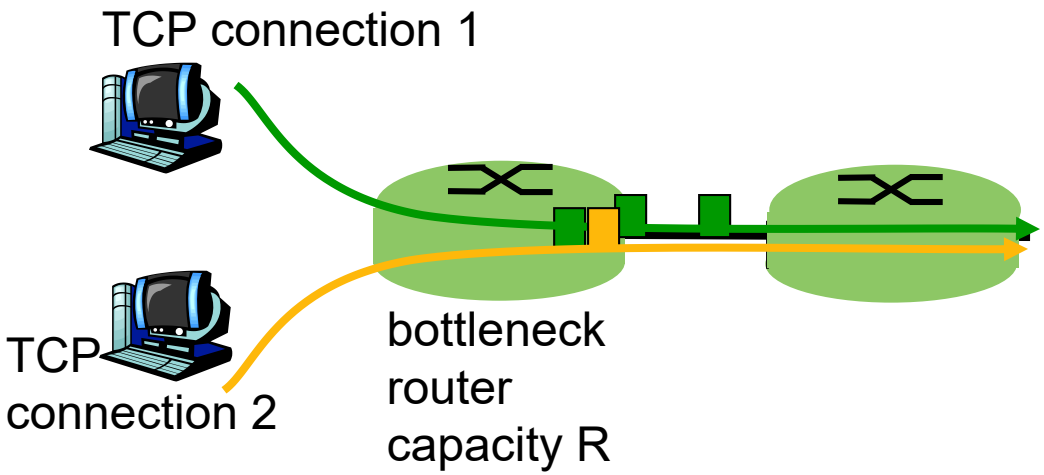
TCP Congestion Control Summary

- When **`CongWin`** is below **`Threshold`**, sender in slow-start phase, window grows exponentially.
- When **`CongWin`** is above **`Threshold`**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** set to **`Threshold`**.
- When timeout occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** is set to 1 MSS.

TCP Fairness

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

- Additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally



Bandwidth Metrics

- Bandwidth Capacity
  - The maximum amount of data per time unit that the link or path has available, when there is no competing traffic
  - The link with the minimum transmission rate determines the capacity
- Achievable bandwidth (Throughput)
  - The maximum amount of data per time unit that a link or path can provide to an application, given the current utilization, the protocol and operating system used, and the end-host performance capability and load
  - Hardware/software configuration on the end hosts actually limit the achievable bandwidth delivered to the application.
- Bandwidth Utilization
  - The aggregate capacity currently being consumed
- Available Bandwidth
  - *Available Bandwidth = Bandwidth Capacity – Bandwidth Utilization*

Passive vs. Active measurement

- Active Measurement
  - Tools actively send probing packets into the network
  - Overhead introduced
- Passive Measurement
  - Tools monitors the passing traffic without interfering
  - Less reliable than active – cannot selectively measure all aspect of bandwidth

Measurement Techniques

- Receiver-based (end-to-end)
  - Usually use the one-direction TCP stream to probe the path bandwidth
  - Advantage
    - More accurate than sender-based technique
  - Disadvantage
    - Difficult to deploy
    - The clock have to be synchronized at two ends
- Sender-based (echo-based)
  - Force the receiver to reply the ICMP query, UDP echo or TCP-FIN
  - Advantage
    - Flexible deployment
    - Clock needn't synchronized at two ends.
  - Disadvantage:
    - ICMP and UDP echo packets are rate-limited or filtered out by some routers
    - Round-trip is influenced by cross-traffic than that of one-way delay
    - Response packets may come back through a different path

Measurement Technique Examples

- Packet Dispersion technology
  - Packet pair and packet train
  - Self-Loading Periodic streams (SLOPS)
- Variable Packet Size (VPS) technology
  - VPS even/odd
  - Tailgating technique
- TCP/UDP/ICMP Simulation

TCP/UDP/ICMP Simulation

Tool Name	Method	Protocol	Metrics	Path/Per-link
TReno	TCP simulation	UDP, ICMP	BTC	Path
ttcp	Path flooding	TCP, UDP	Achievable bandwidth	Path
iperf	**Path flooding**	**TCP, UDP**	**Bandwidth capacity, Loss**	**Path**
Netperf	Path flooding	TCP, UDP	BTC, delay throughput	Path

FOR MORE INFO...

TReno http://www.psc.edu/networking/treno_info.html

Iperf http://dast.nlanr.net/Project/Iperf

Netperf http://www.netperf.org/netperf/NetperfPage.html

ttcp ftp://ftp.arl.mil/pub/ttcp/

IPERF

- Website
  - http://dast.nlanr.net/Projects/Iperf/
- Format
  - server side
    - iperf -s -%
    - Iperf -s -V
  - client side
    - iperf -c <server address> -%
    - iperf -c <server IPv6 address>

Variable Packet Size Technology

Tool Name	Protocol	Metrics	Path/Per-link
bing	ICMP	Bandwidth capacity, loss, delay	Path
clink	UDP	Bandwidth capacity, Loss	Path
Pchar	UDP, ICMP	Bandwidth capacity, Loss, delay	Per-link
Nettimer	TCP	Bandwidth capacity	Per-link
pathchar	UDP, ICMP	Bandwidth capacity, Loss, delay	Per-link

FOR MORE INFO...

Bing http://www.cnam.fn/reseau/bing.html
Clink http://rocky.wellesley.edu/downey/clink/
Pchar http://www.emplyees.org/~bmah/software/pchar
Nettimer http://mosquitonet.stanford.edu/~laik/project/nettimer
Pathchar ftp://ftp.ee.lbl.gov/pathchar/

Packet Dispersion Technique

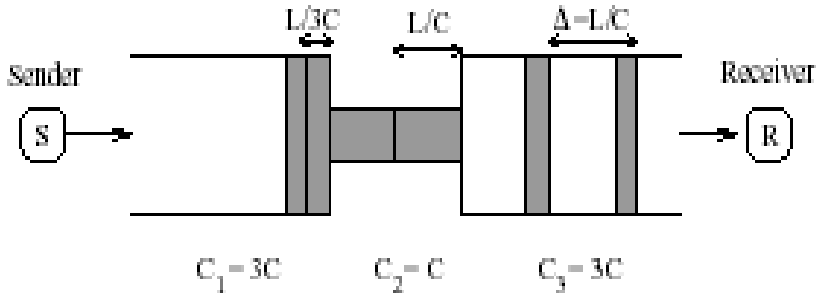
Tool Name	Method	Protocol	Metrics
bprobe	Packet pair	ICMP	Bandwidth Capacity
cprobe	Packet pair	ICMP	Bandwidth utilization
Netest	Packet pair	UDP	Bandwidth capacity
Pathrate	Packet pair, packet train	UDP	Bandwidth capacity
Pipechar	Packet train	UDP	Available bandwidth
Sprobe	Packet pair	TCP	Bandwidth capacity

FOR MORE INFO...

Bprobe and cprobe http://cs-people.bu.edu/carter/tools/Tools.html
Nettest http://www-didc.lbl.gov/pipechar
Pathrate http://www.cc.gatech.edu/fac/Constantinos.Dovrolis
Pipechar http://www-didc.lbl.gov/pipechar
SProbe http://sprobe.cs.washington.edu

Packet Dispersion Technique

- Sender sends two same-size packets back-to-back from source to sink.
- The packets will reach the sink dispersed by the transmission delay of the bottleneck links if there is no cross traffic
- Measuring the dispersion can infer the bottleneck link bandwidth capacity.



Note: Bottleneck link can refer to the link with smallest transmission rate, it's also can refer to the link with minimum available bandwidth. We refer the bottleneck link to the first case.