CSC4200/5200 – COMPUTER NETWORKING

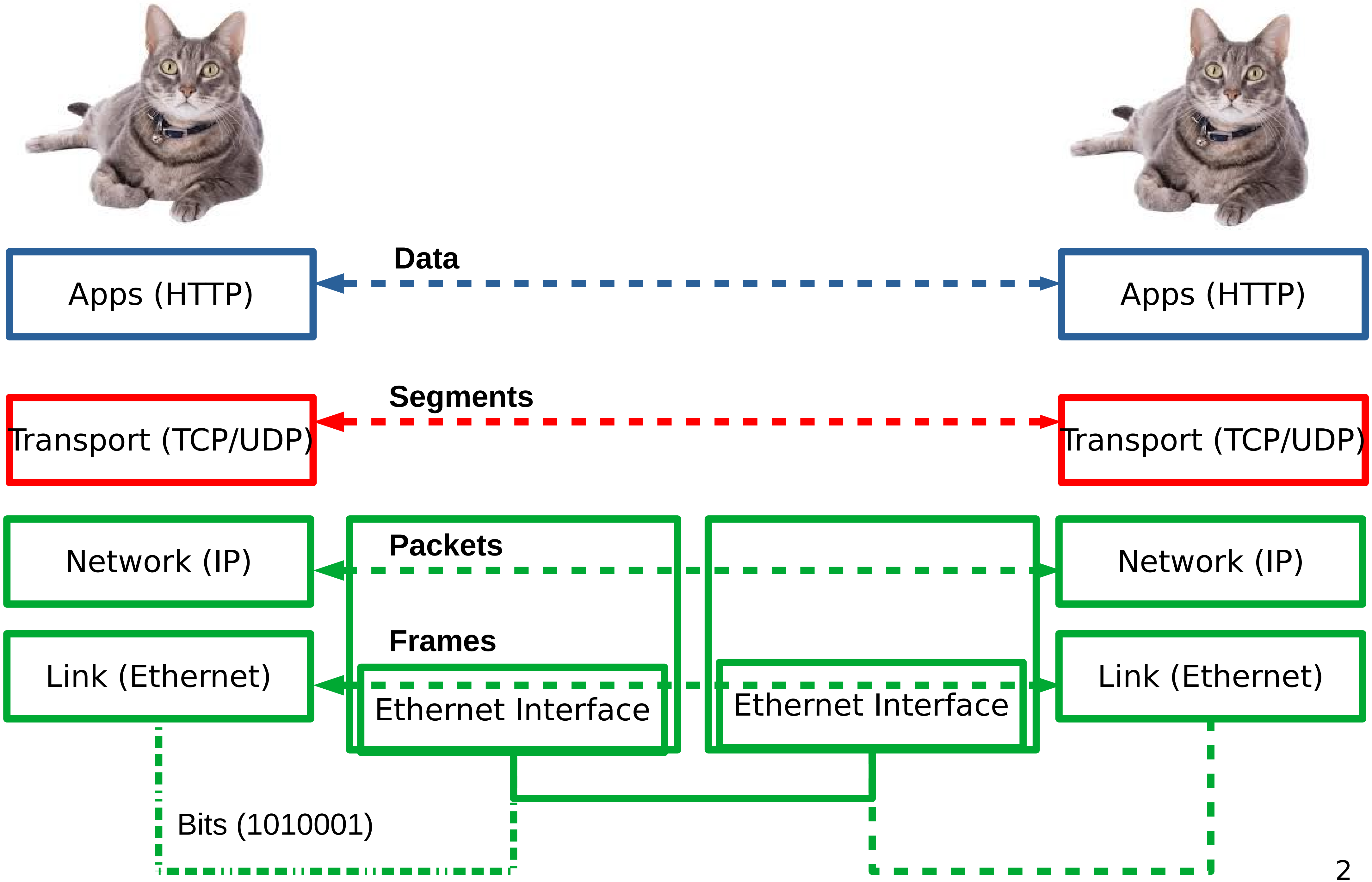
Instructor: Susmit Shannigrahi

TRANSPORT LAYER PROTOCOLS

sshannigrahi@tntech.edu

GTA: dereddick42@students.tntech.edu

Tennessee TECH



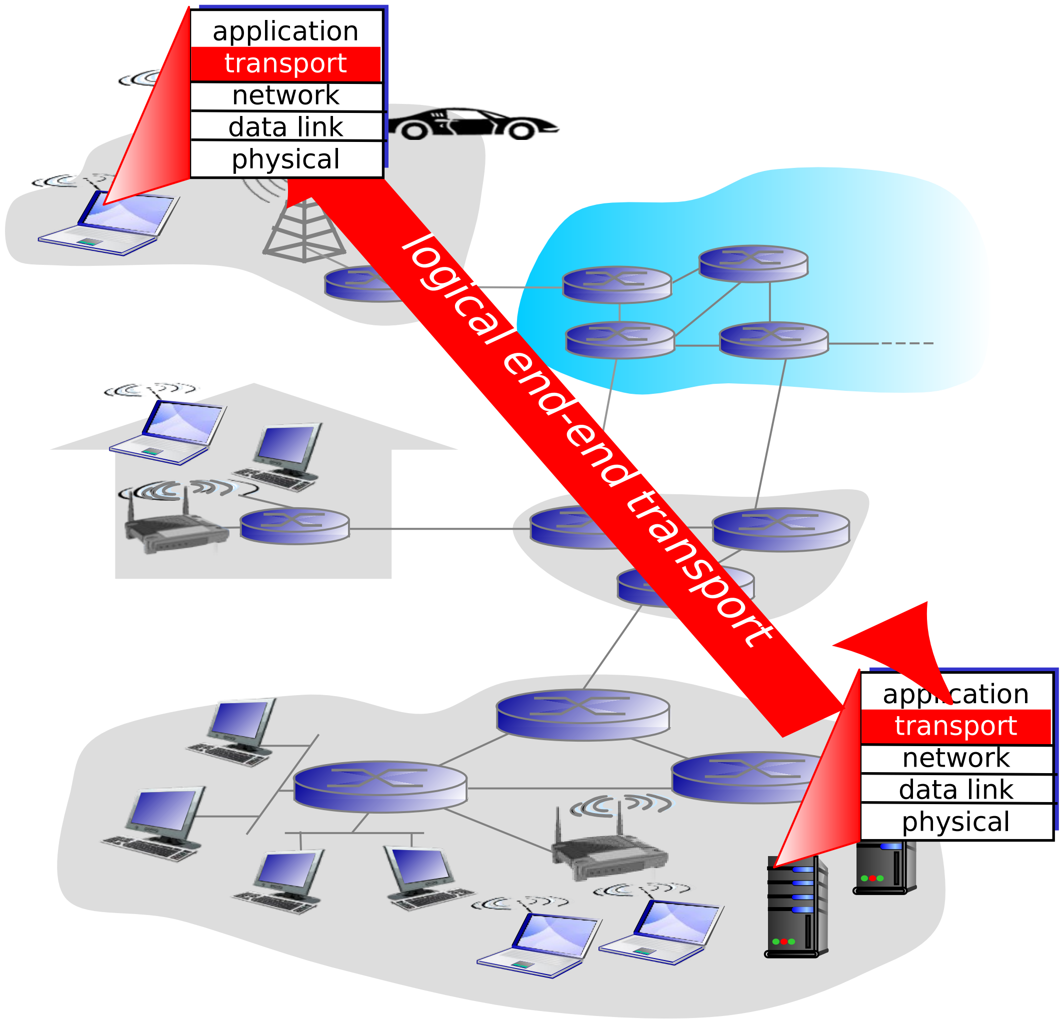
Data
Apps (HTTP)
Apps (HTTP)
Segments
Transport (TCP/UDP)
Transport (TCP/UDP)
Packets
Network (IP)
Network (IP)
Frames
Link (Ethernet)
Ethernet Interface
Ethernet Interface
Link (Ethernet)
Bits (1010001)

What is transport layer?

- Problem: How to turn this host-to-host packet delivery service into a process-to-process communication channel?

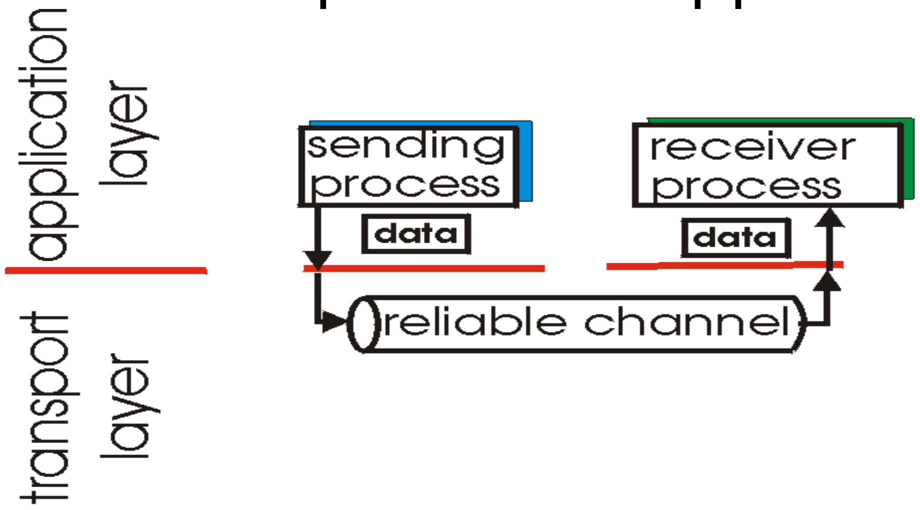
Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into *segments*, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Principles of reliable data transfer

- important in application, transport, link layers

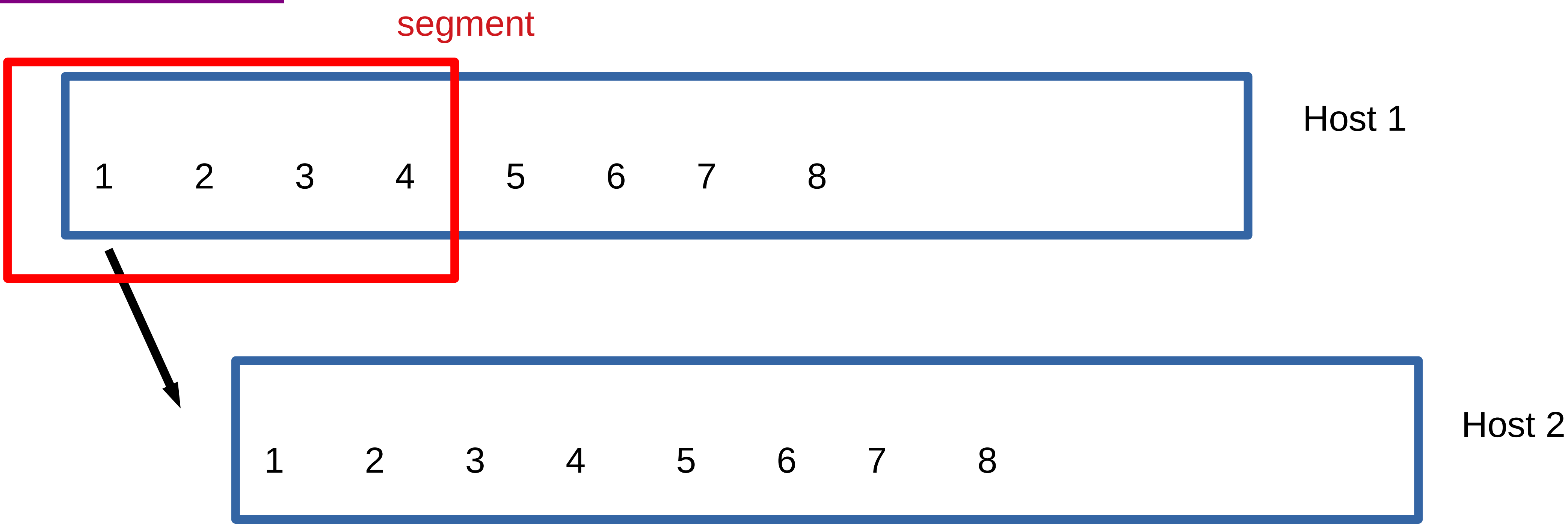


(a) provided service

TCP – Transmission Control Protocol

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

TCP – Transmission Control Protocol



TCP Segment

IP → No more than MTU (1500 Bytes)

TCP header → 20 bytes

TCP segment → 1460 bytes

Why?

TCP Header

SYN

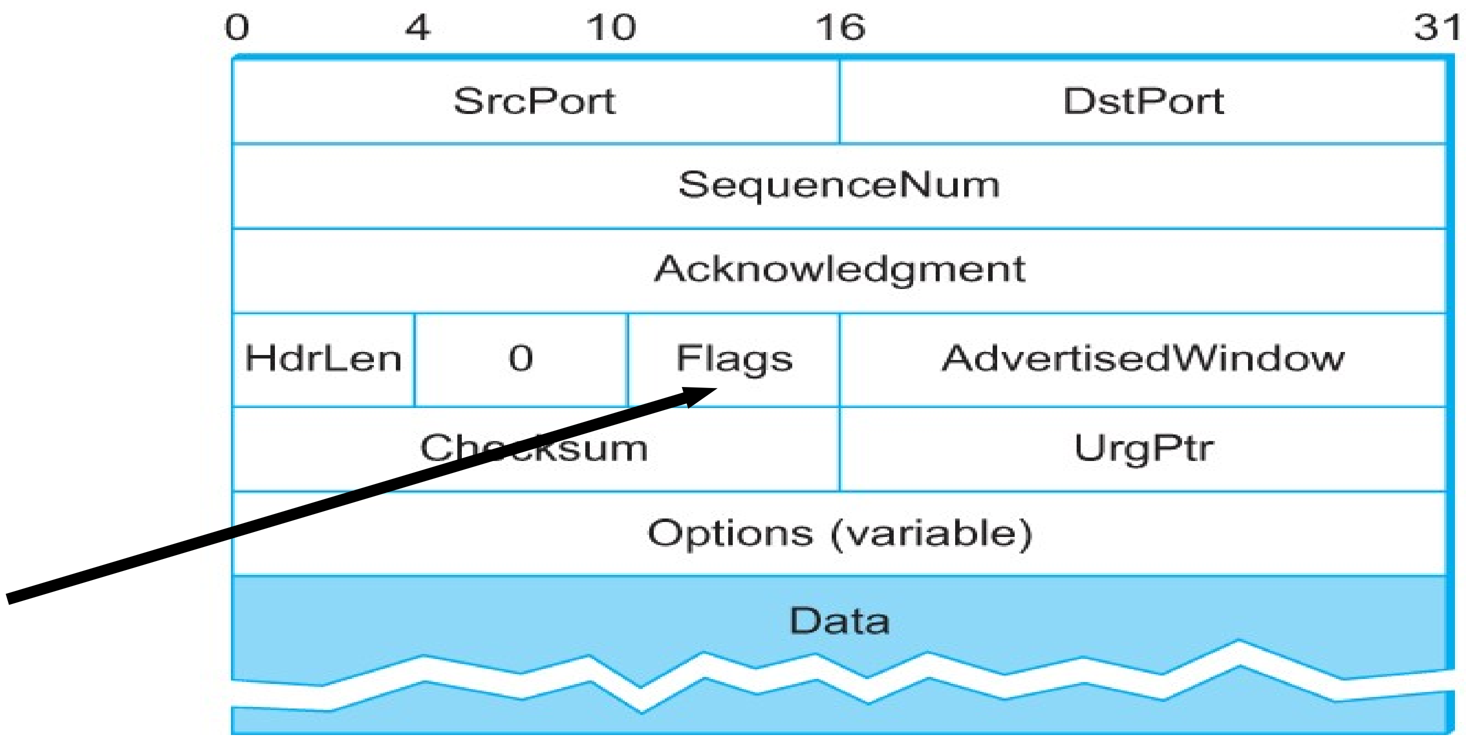
FIN

RST

PSH

URG

ACK

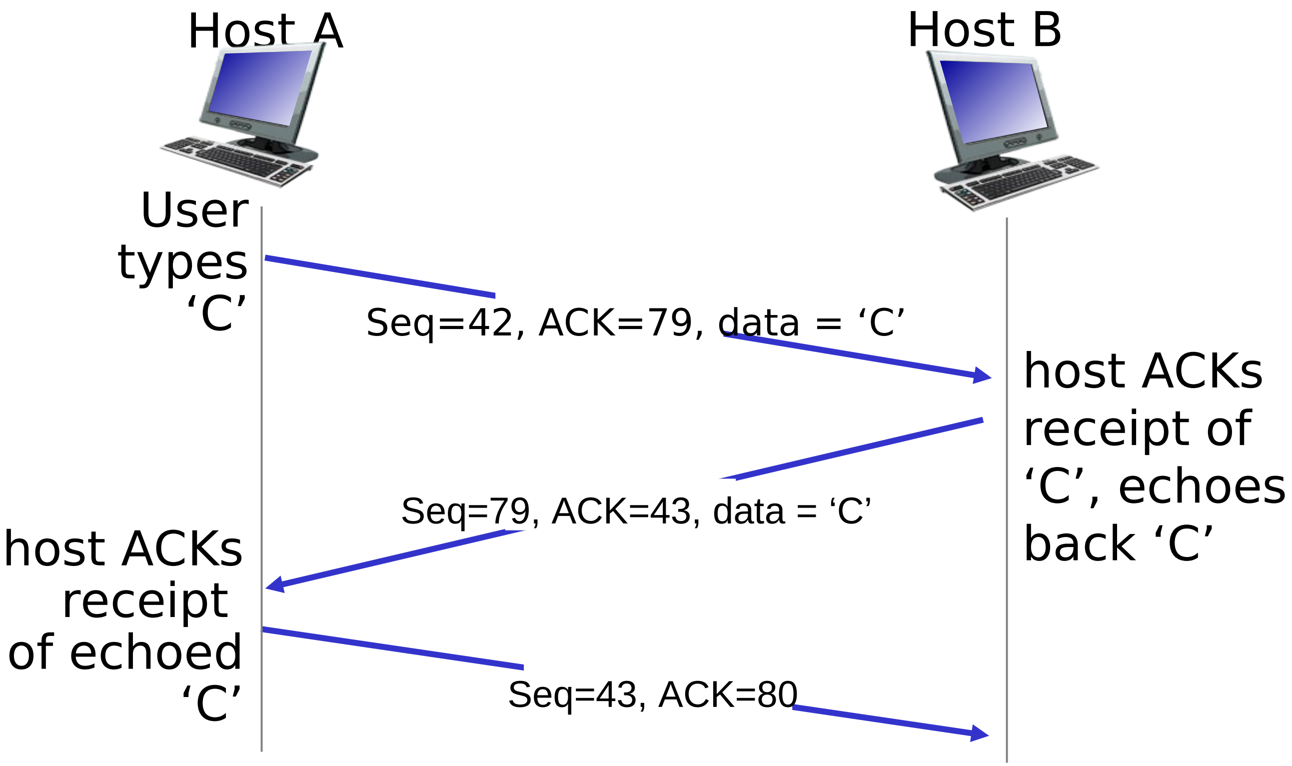


TCP Header Format

TCP – Transmission Control Protocol

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

TCP seq. numbers, ISNs



simple telnet scenario

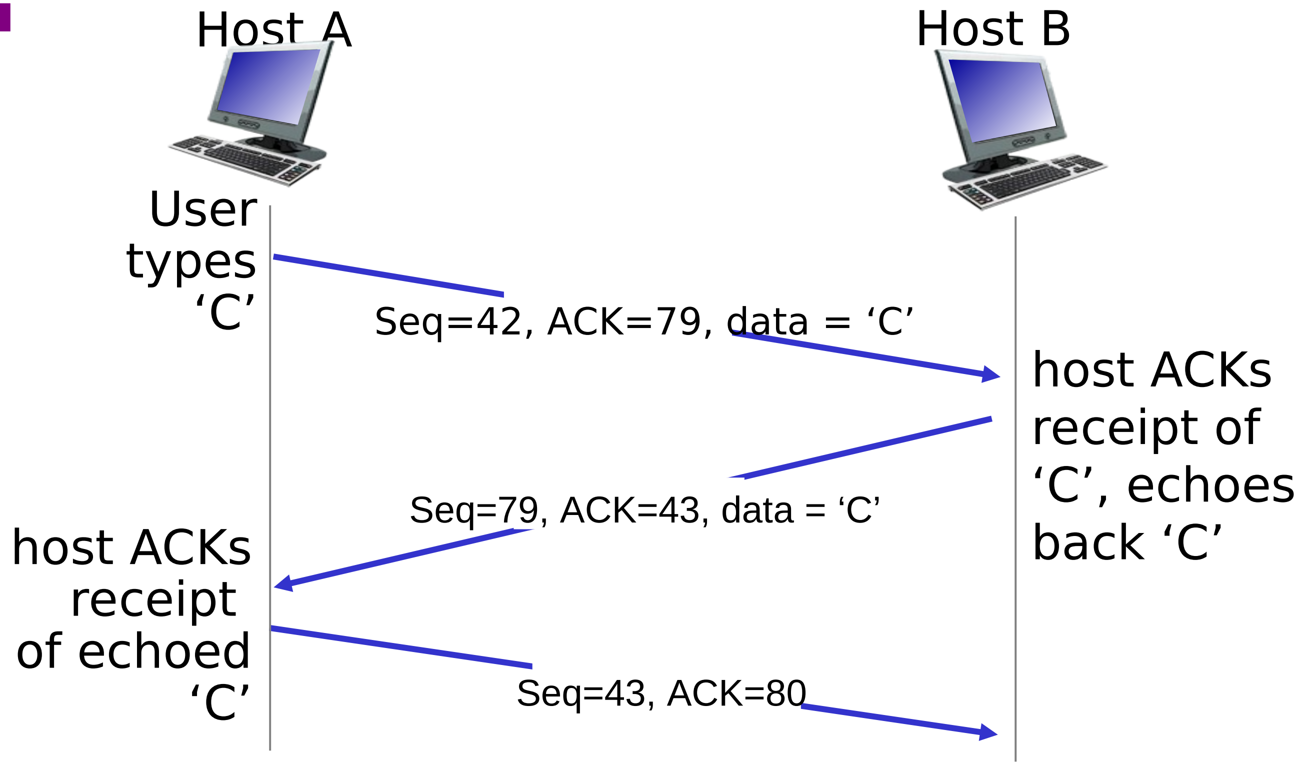
Sequence number for the first byte

Why not use 0 all the time?

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

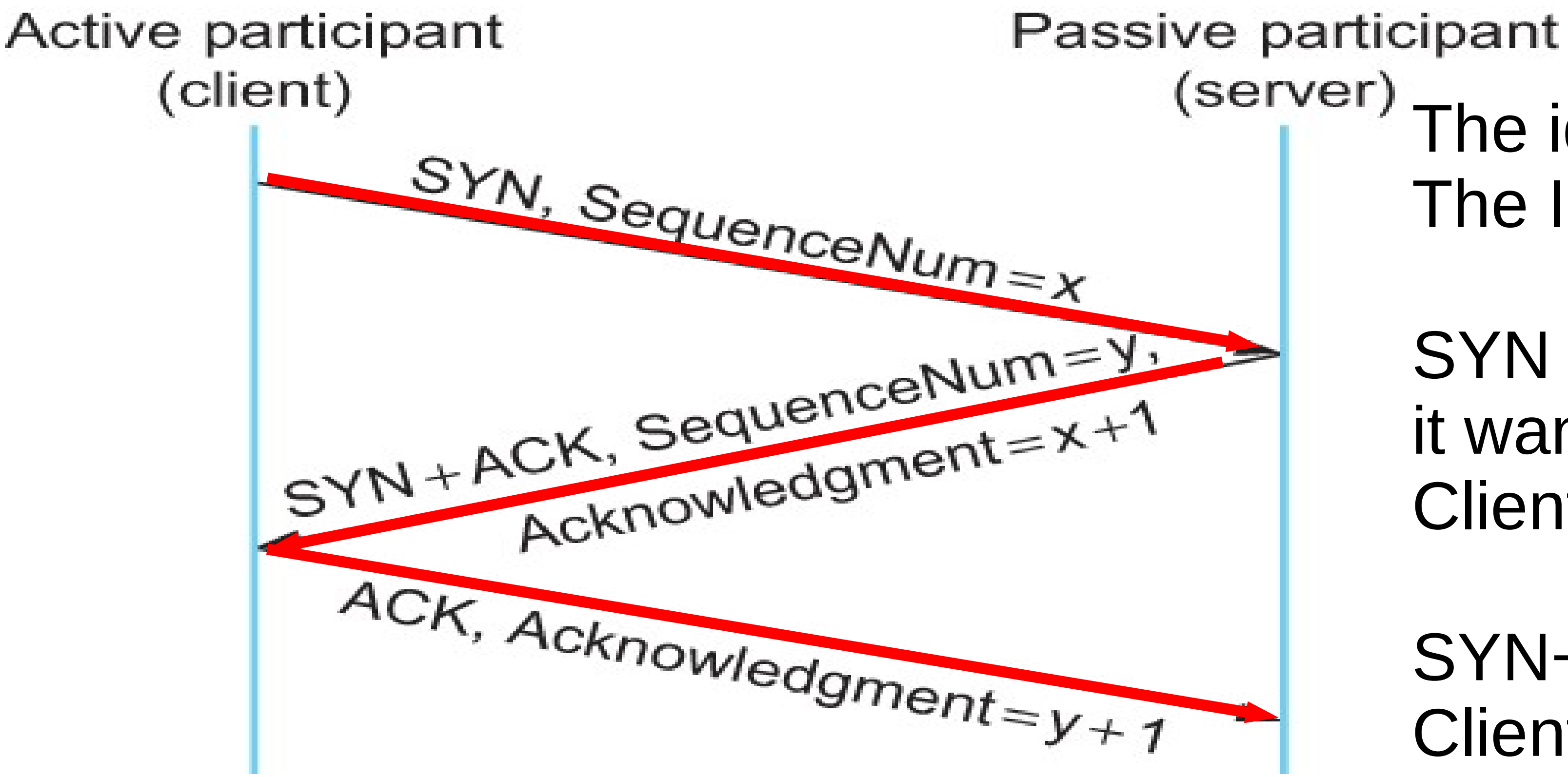
- TCP ISNs are clock based
  - 32 bits, increments in 4 microseconds
  - 4.55 hours wrap around time

TCP seq. numbers, ACKs



simple telnet scenario

TCP Three-way Handshake



Timeline for three-way handshake algorithm

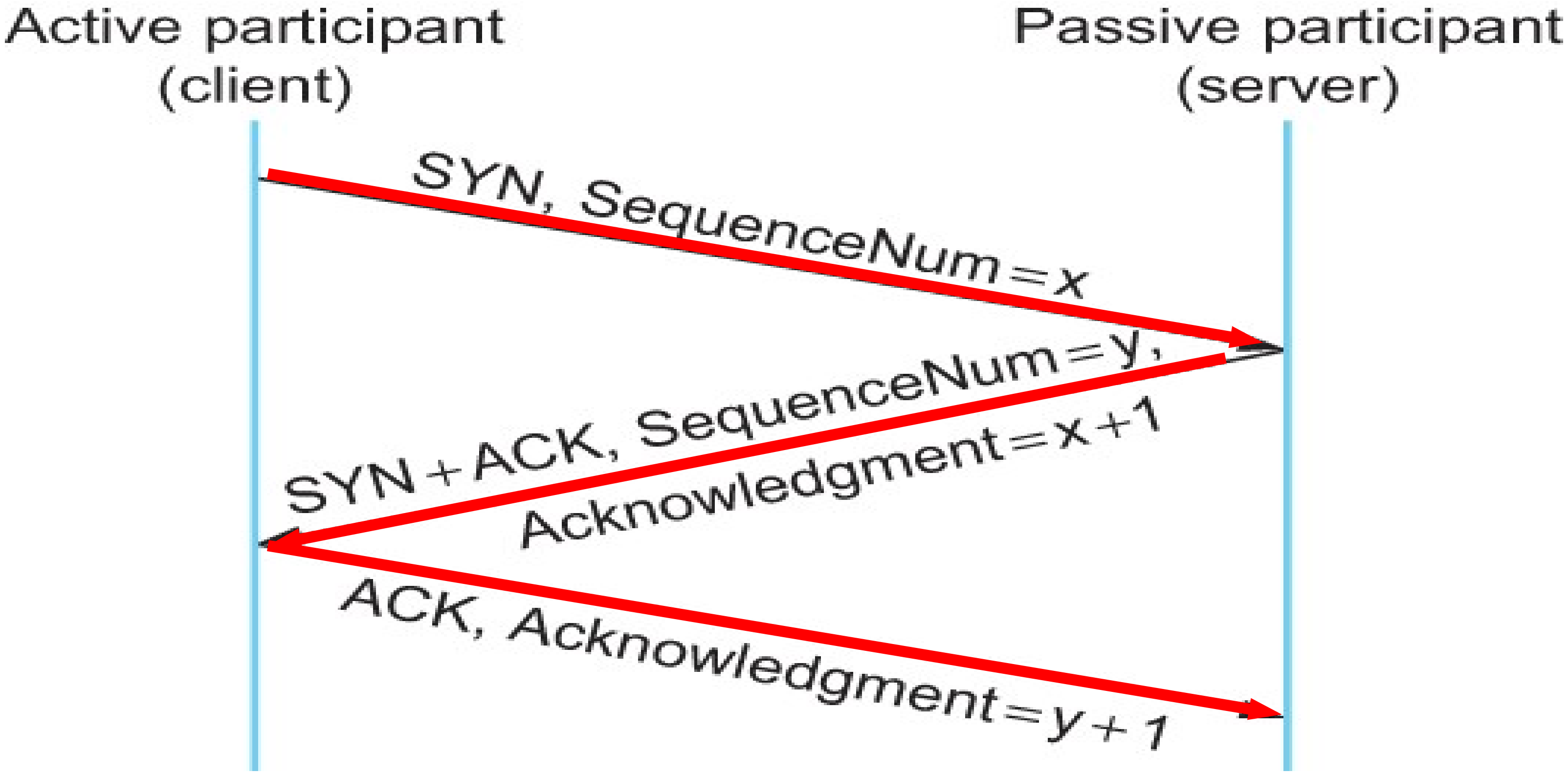
The idea is to tell each other
The ISNs

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?

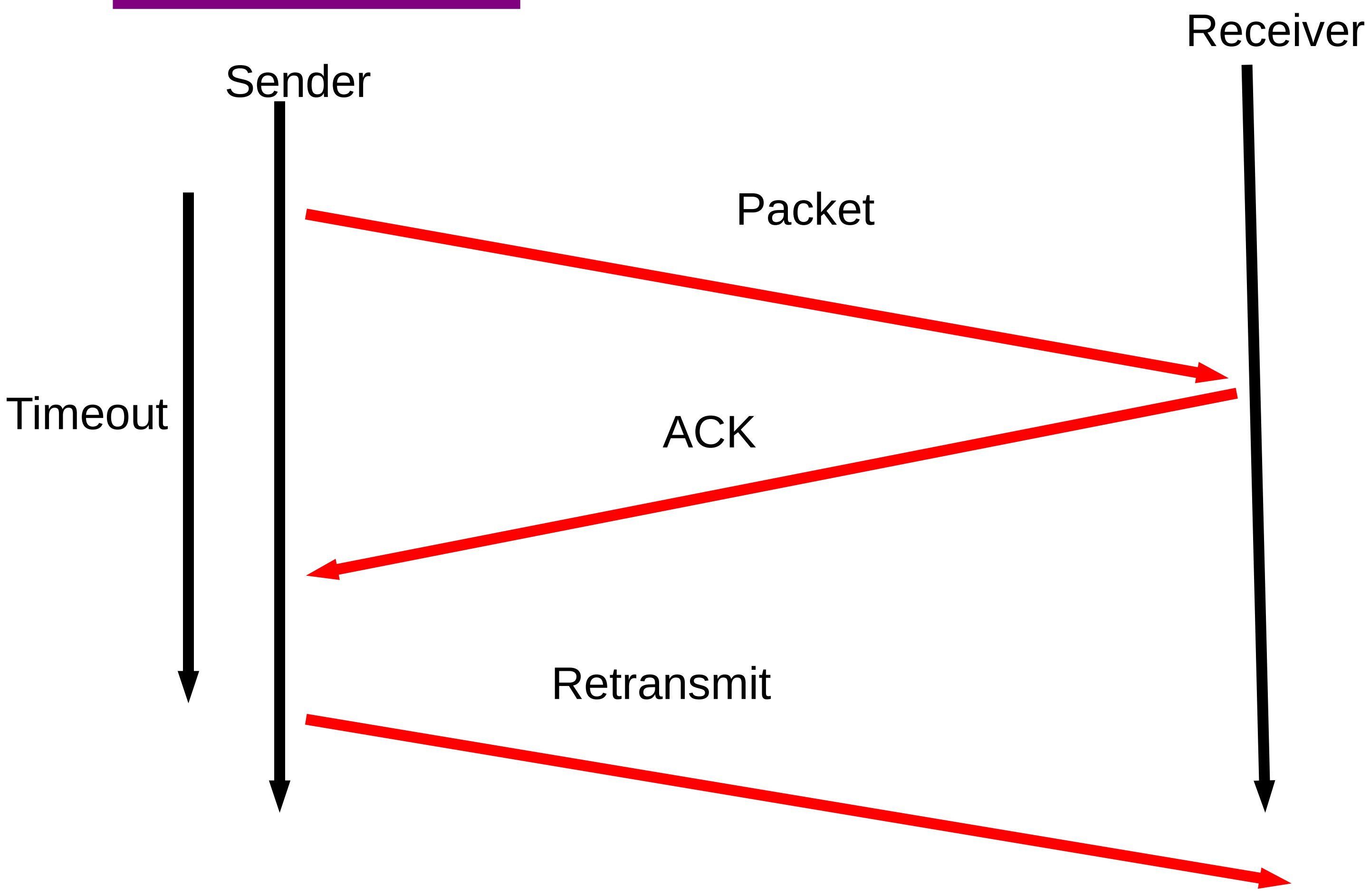
What if the SYN is lost?



Start Timer and resend

Timeline for three-way handshake algorithm

TCP Retransmission - ARQ

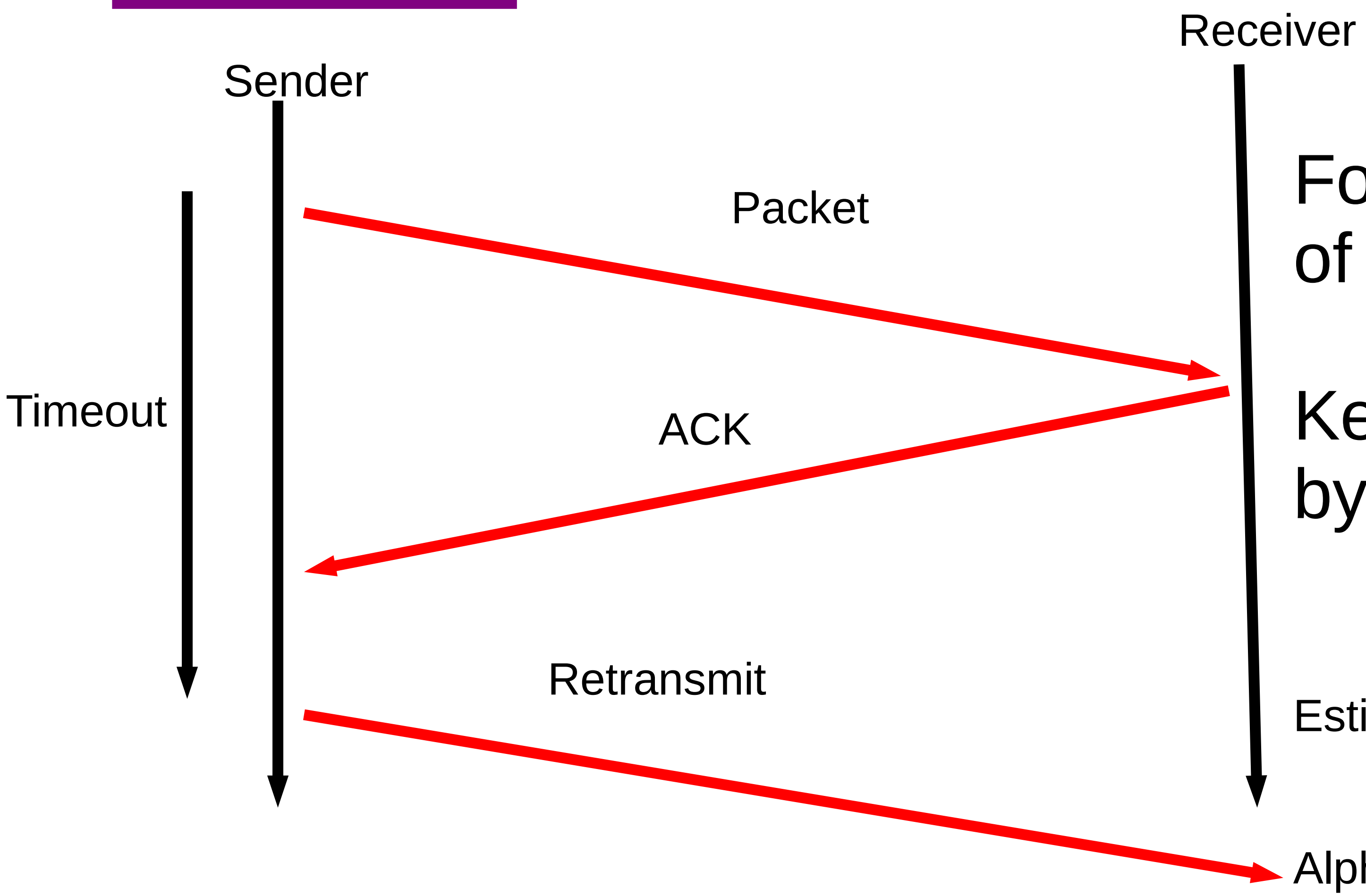


Each packet is “ACK”ed by the receiver

If ACK isn’t received by timeout, resend

Example, Stop-n-wait

How long should the sender wait?



For TCP, timeout is a function of RTT

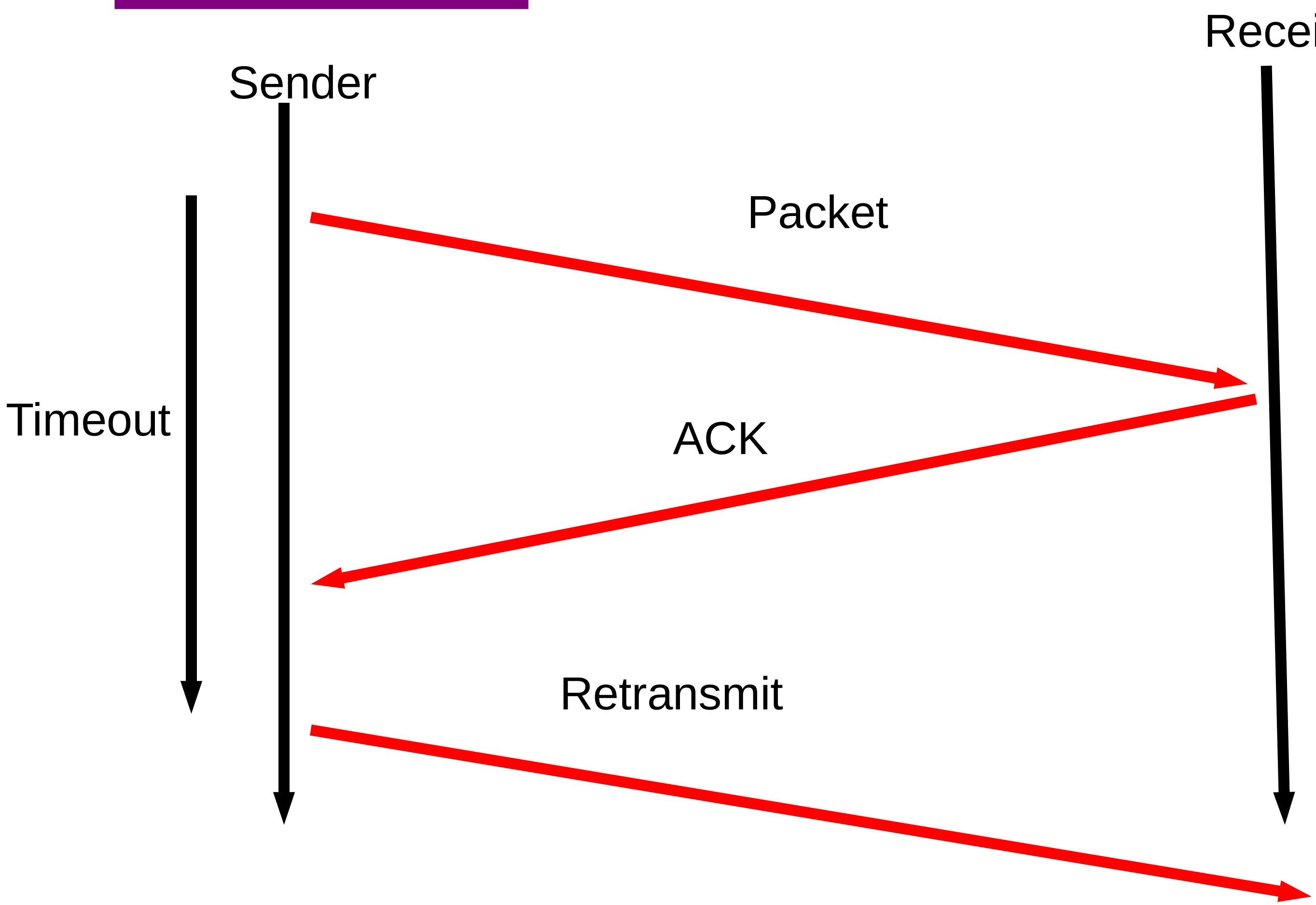
Keep a running estimate of RTTs by watching the ACKs

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

Alpha is generally 0.125

Timeout = 2* EstimatedRTT

But stop and wait is inefficient

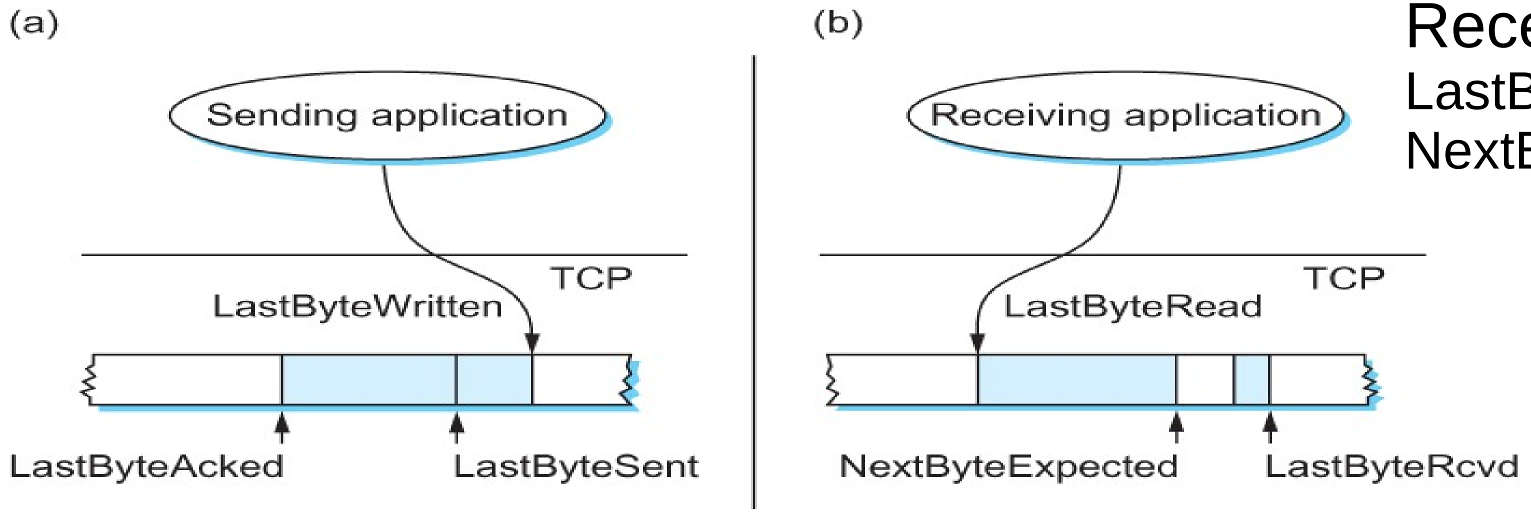


Only one segment in-flight

Especially bad if delay is high!

Solution – sliding window

Sliding Window Revisited



Sending Side

LastByteAcked ≤ LastByteSent

LastByteSent ≤ LastByteWritten

Receiving Side

LastByteRead < NextByteExpected

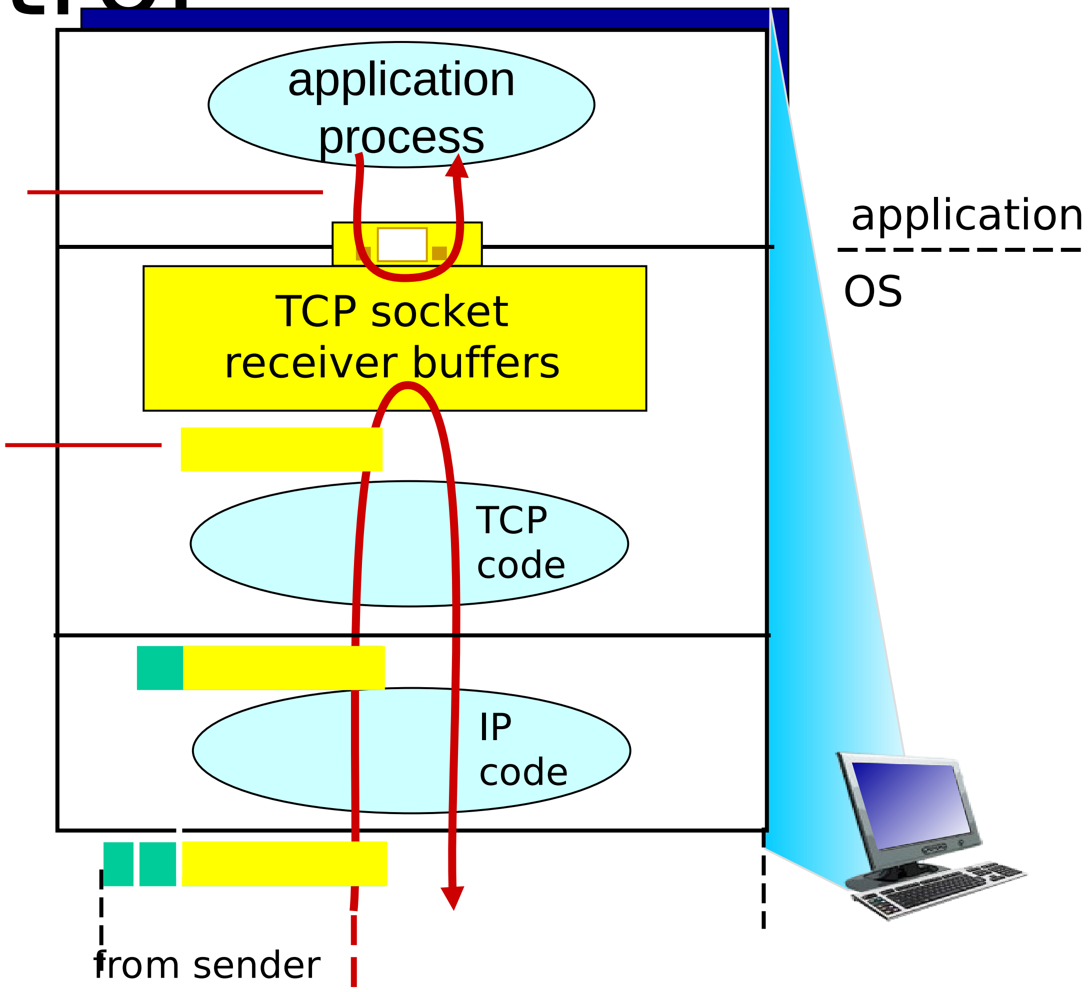
NextByteExpected ≤ LastByteRcvd + 1

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

Used for TCP flow control

application may remove data from TCP socket buffers ….

… slower than TCP receiver is delivering (sender is sending)



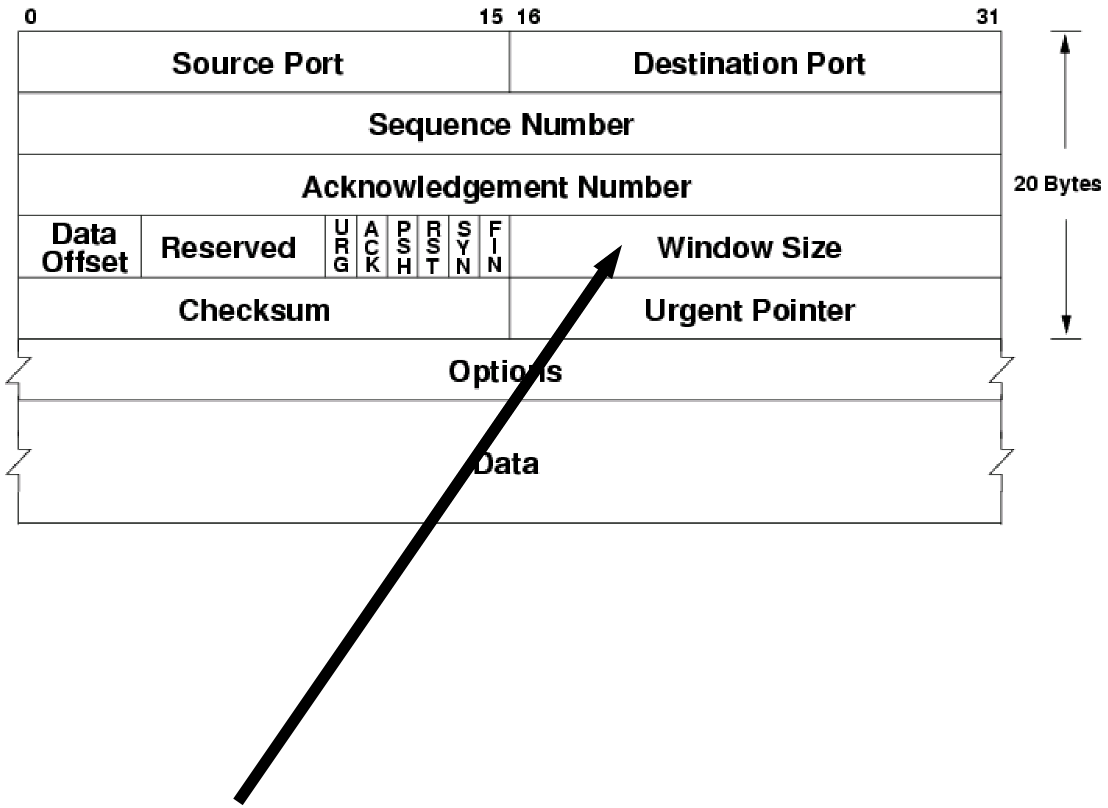
receiver protocol stack

flow control

receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast

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

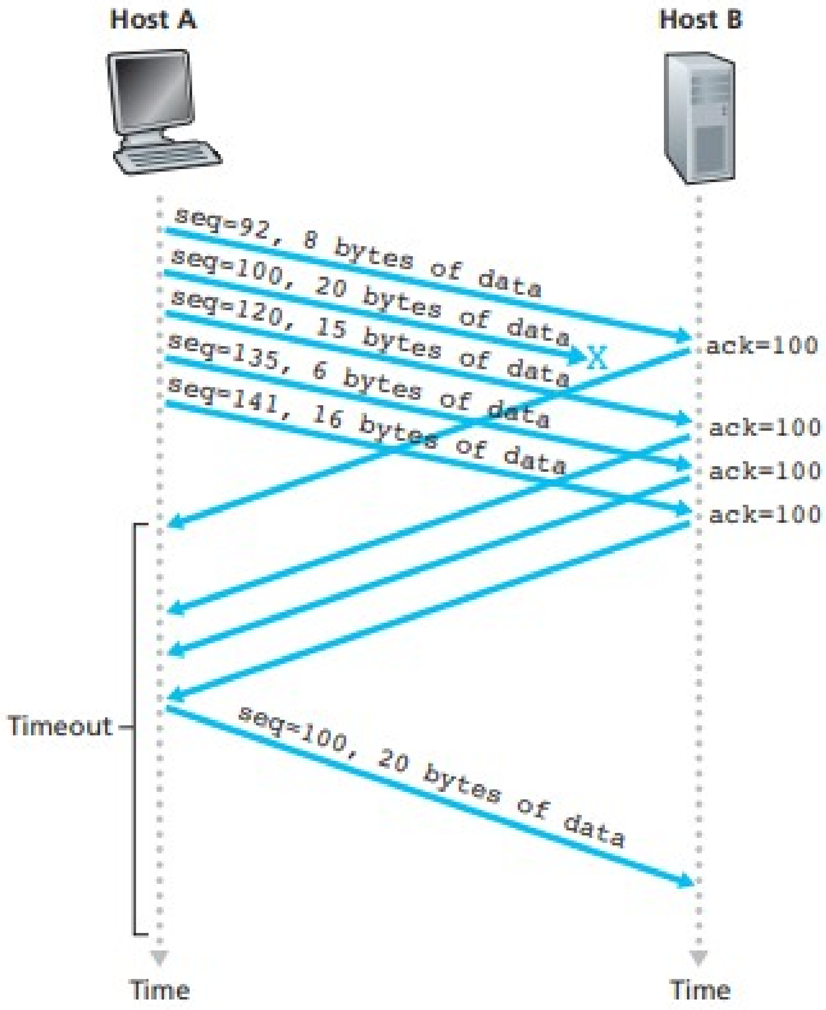


TCP Fast Retransmission

Timeouts are wasteful

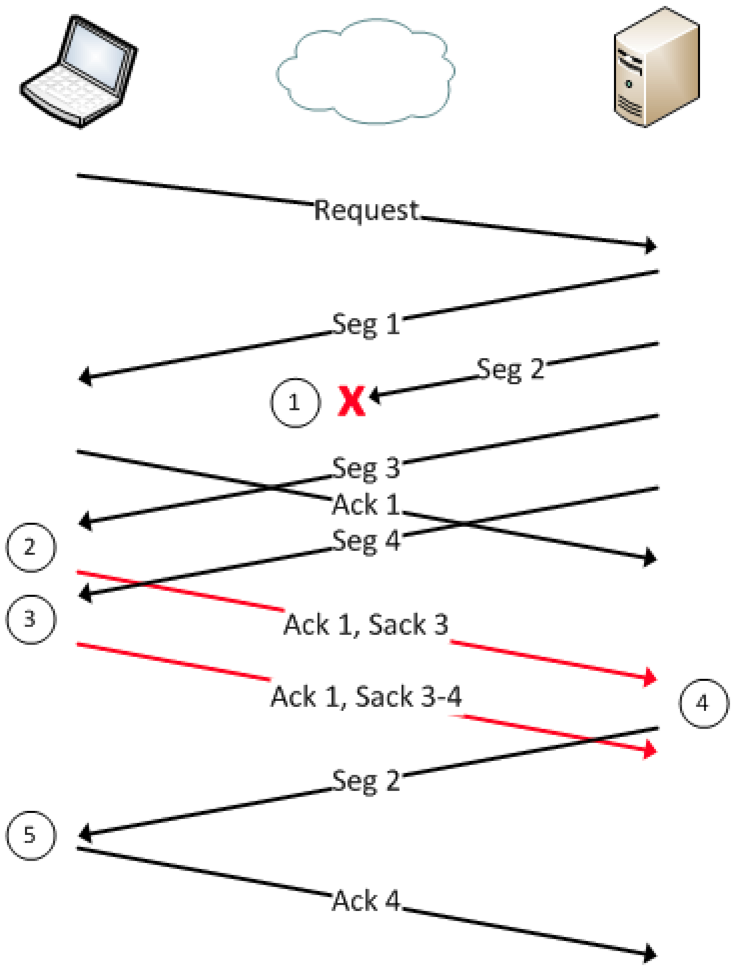
Triple duplicate ACKs

Retransmits before timeout



TCP Fast Retransmission - SACK

What if multiple segments are lost?

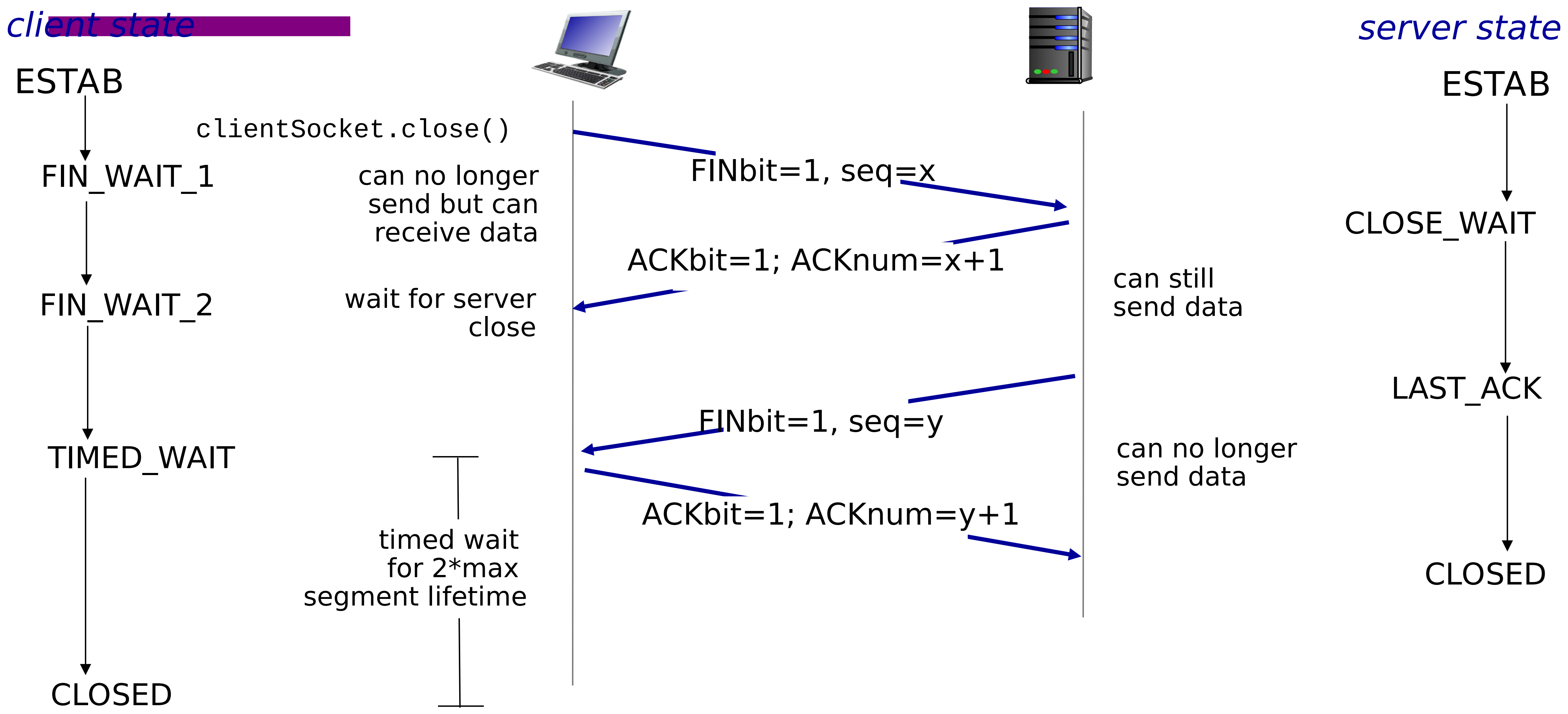


Very good explanation:
https://packetlife.net/blog/2010/jun/17/tcp-selective-acknowledgments-sack/

TCP: closing a connection

- client, server each close their side of connection
  - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
  - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

TCP: closing a connection



Why do we need ack for closing?

- Data in-flight

Congestion Control

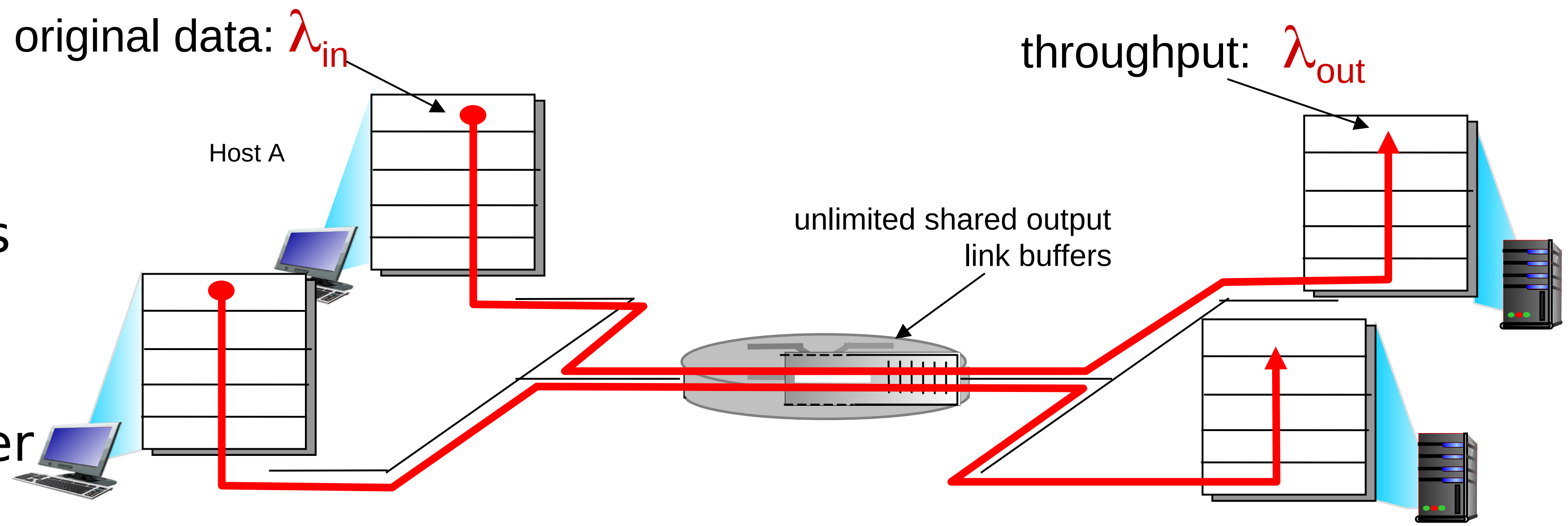
Principles of congestion control

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!

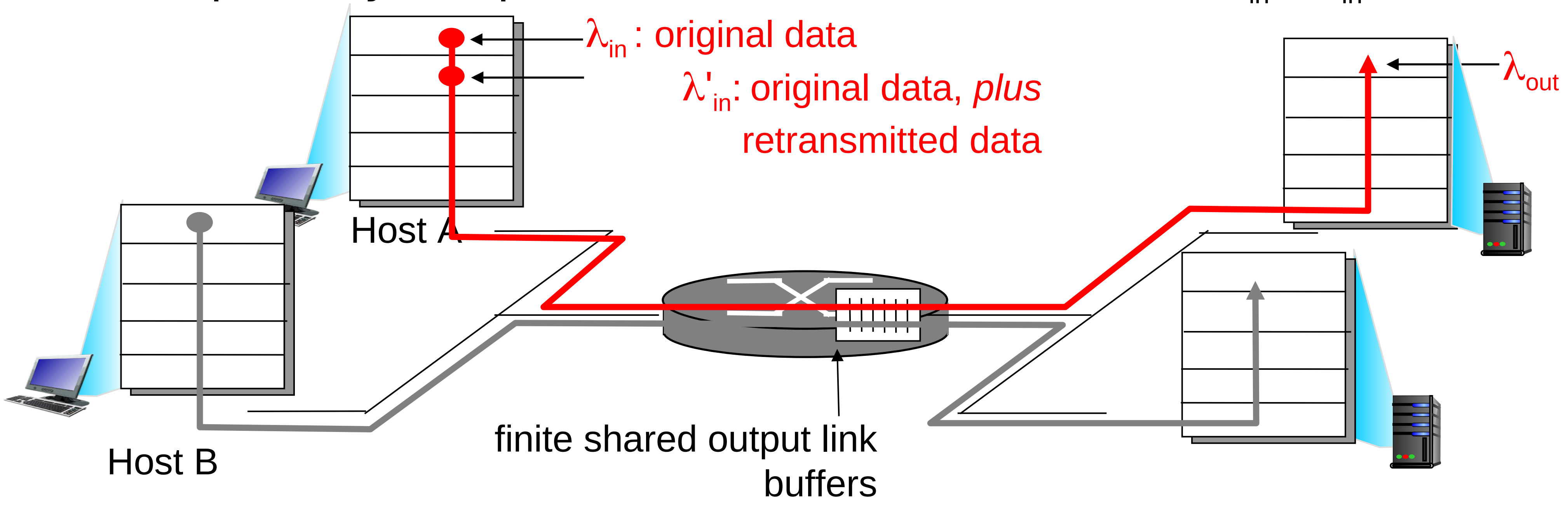
Congestion: scenario 1

- 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

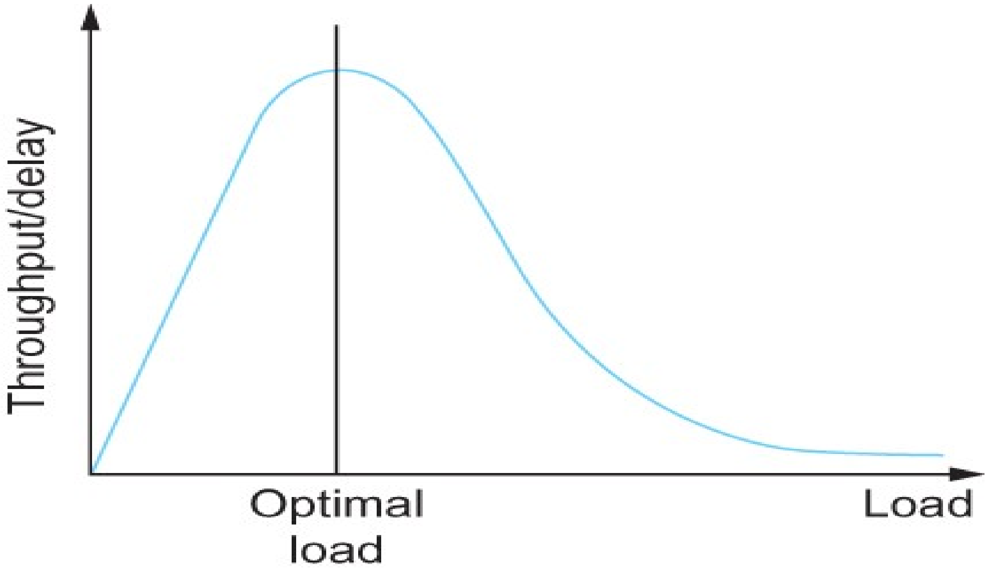
High throughput –

- 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

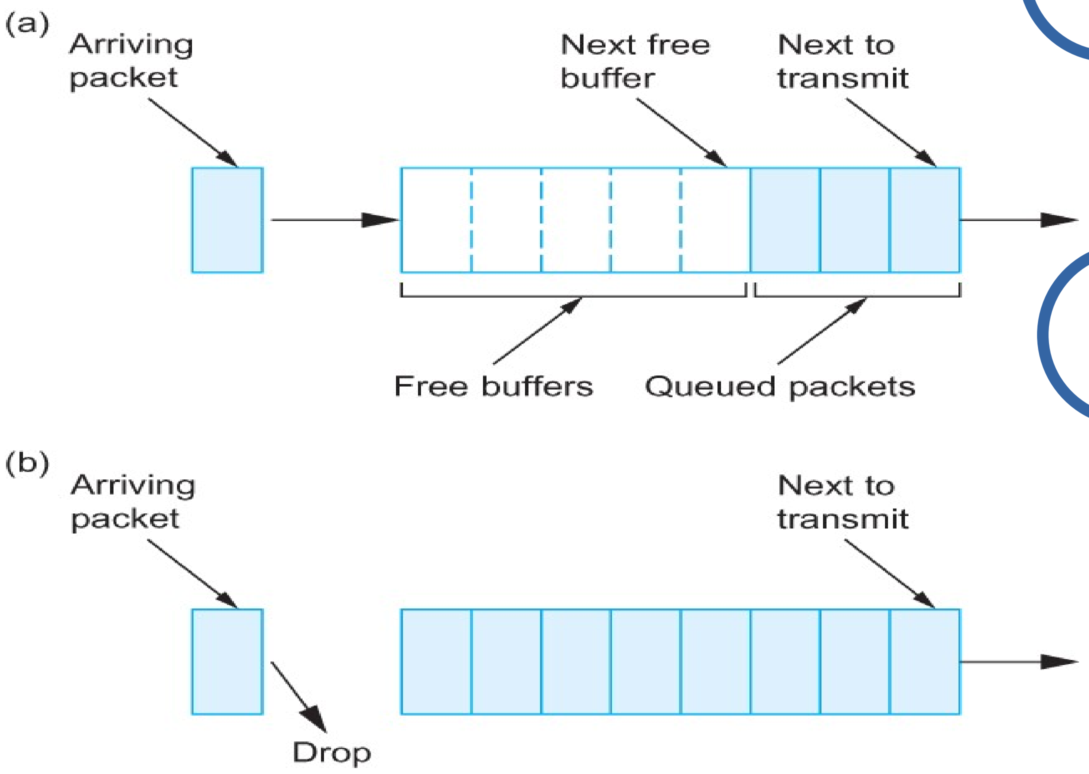
- Evaluation Criteria
  - Fair Resource Allocation
    - The effective utilization of network resources is not the only criterion for judging a resource allocation scheme.
    - We want to be “fair”
    - Equal share of bandwidth

But, what if the flows traverse different paths?

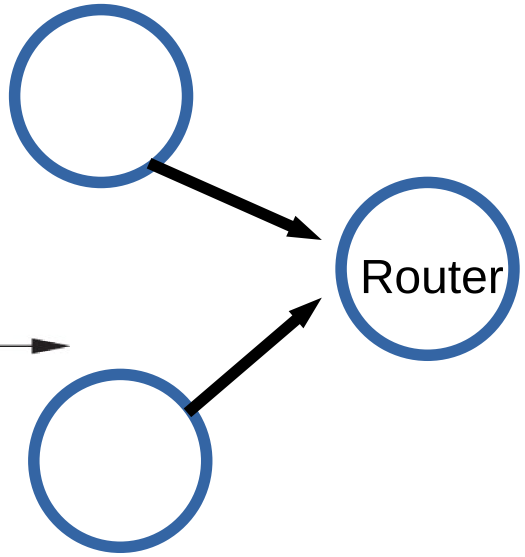
Open problem, often determined by economics

Queuing Disciplines

Simplest – FIFO and drop tail



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



What are the problems?

Defining Fairness: Flows

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

What is a flow?

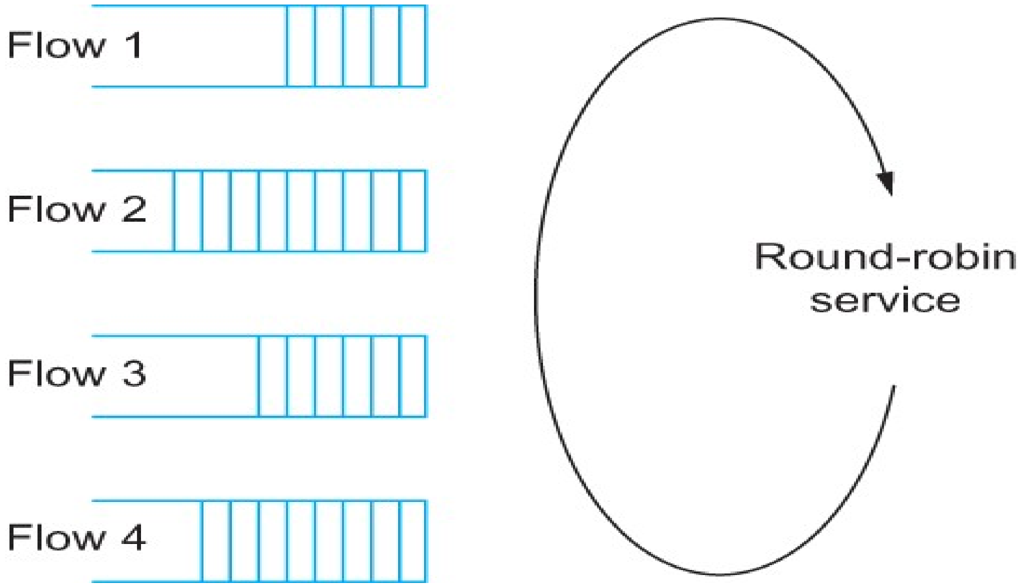
Combination of <Src IP, Src Port, Dst IP, Dst Port>

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

Reading

https://book.systemsapproach.org/e2e/tcp.html#segment-format
https://book.systemsapproach.org/e2e/tcp.html#connection-establishment-and-termination
https://book.systemsapproach.org/e2e/tcp.html#sliding-window-revisited