

COMPUTER NETWORKS LAB (CS315)

Assignment-5 TCP

Date: 31 Jan 2023



Indian Institute of Technology Dharwad
भारतीय प्रौद्योगिकी संस्थान धारवाड़

TCP: overview

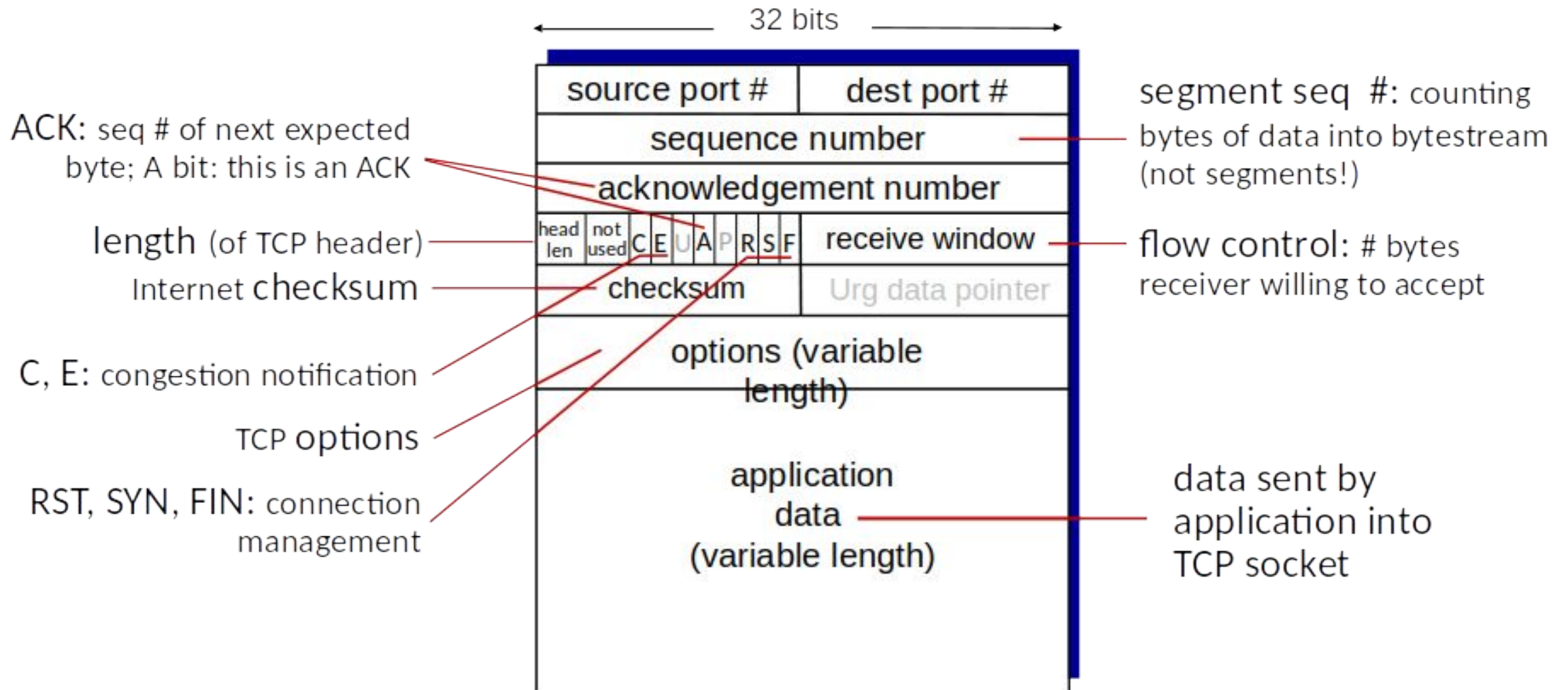
RFCs: 793, 1122, 2018, 5681, 7323

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

Transport Layer: 3-<number>



TCP segment structure



Transport Layer: 3-<number>

TCP sequence numbers, ACKs

Sequence numbers:

- byte stream “number” of first byte in segment’s data

Acknowledgments:

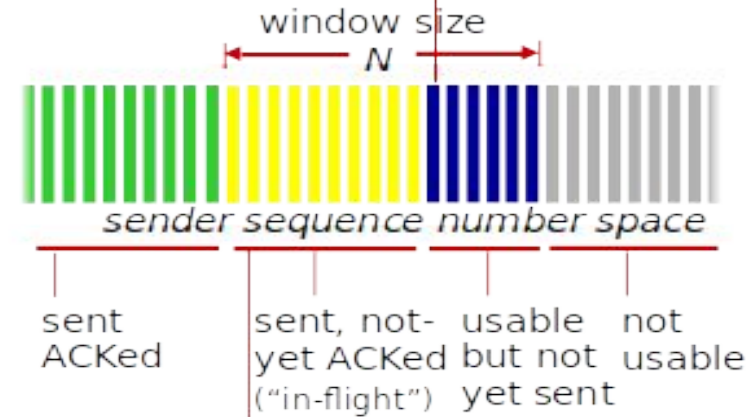
- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A:** TCP spec doesn’t say, - up to implementer

outgoing segment from sender

source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer

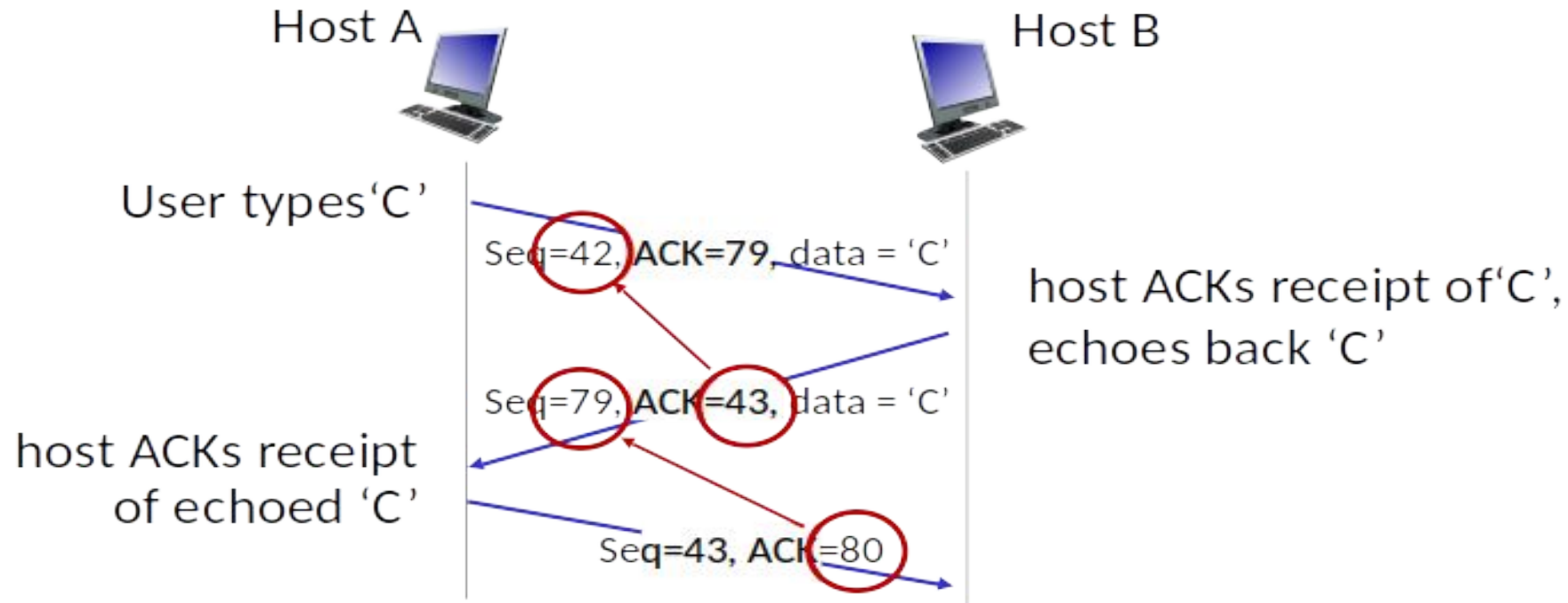


outgoing segment from receiver

source port #	dest port #
sequence number	
acknowledgement number	
	A
	rwnd
checksum	urg pointer

Transport Layer: 3-<number>

TCP sequence numbers, ACKs



simple telnet scenario

TCP round trip time, timeout

Q: how to set TCP timeout value?

- longer than RTT, but RTT varies!
- *too short*: premature timeout, unnecessary retransmissions
- *too long*: slow reaction to segment loss

Q: how to estimate RTT?

- **SampleRTT**: measured time from segment transmission until ACK receipt
 - ignore retransmissions
- **SampleRTT** will vary, want estimated RTT “smoother”
 - average several *recent* measurements, not just current **SampleRTT**

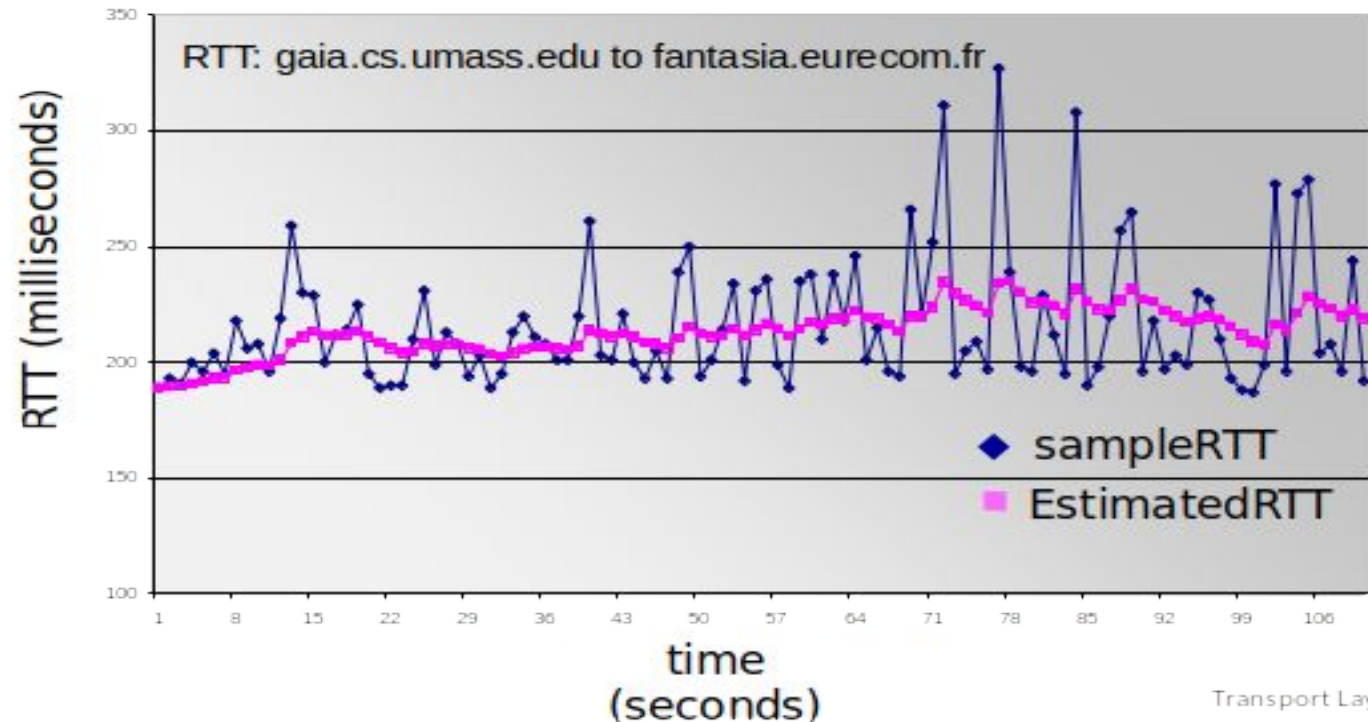
Transport Layer: 3-<number>



TCP round trip time, timeout

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

- exponential weighted moving average (EWMA)
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$



TCP round trip time, timeout

- timeout interval: **EstimatedRTT** plus “safety margin”
 - large variation in **EstimatedRTT**: want a larger safety margin

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$



↑
estimated RTT

↑
“safety margin”

- **DevRTT**: EWMA of **SampleRTT** deviation from **EstimatedRTT**:

$$\text{DevRTT} = (1 - \alpha) * \text{DevRTT} + \alpha * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(typically, $\alpha = 0.25$)

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Transport Layer: 3-85



TCP Sender (simplified)

event: data received from application

- create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running
 - think of timer as for oldest unACKed segment
 - expiration interval: **TimeoutInterval**

event: timeout

- retransmit segment that caused timeout
- restart timer

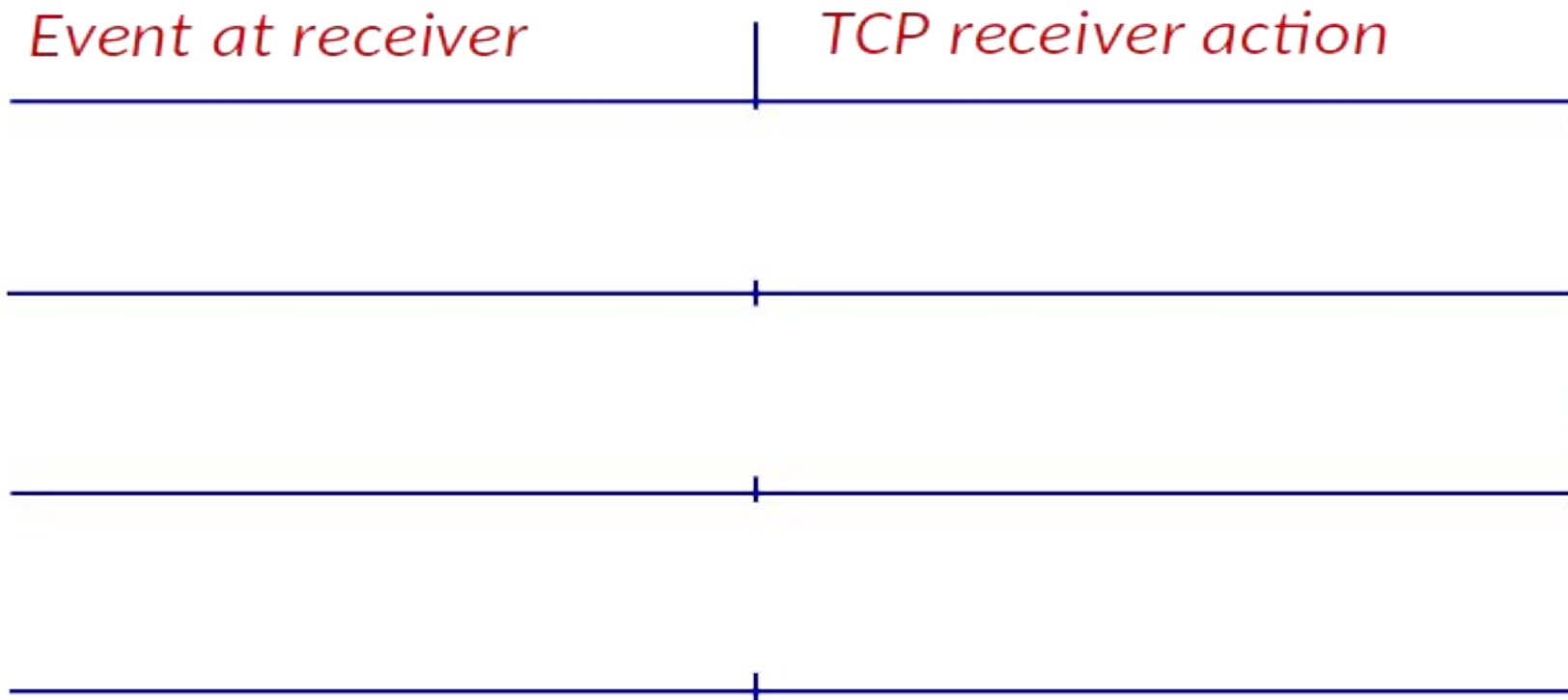
event: ACK received

- if ACK acknowledges previously unACKed segments
 - update what is known to be ACKed
 - start timer if there are still unACKed segments

Transport Layer: 3-<number>

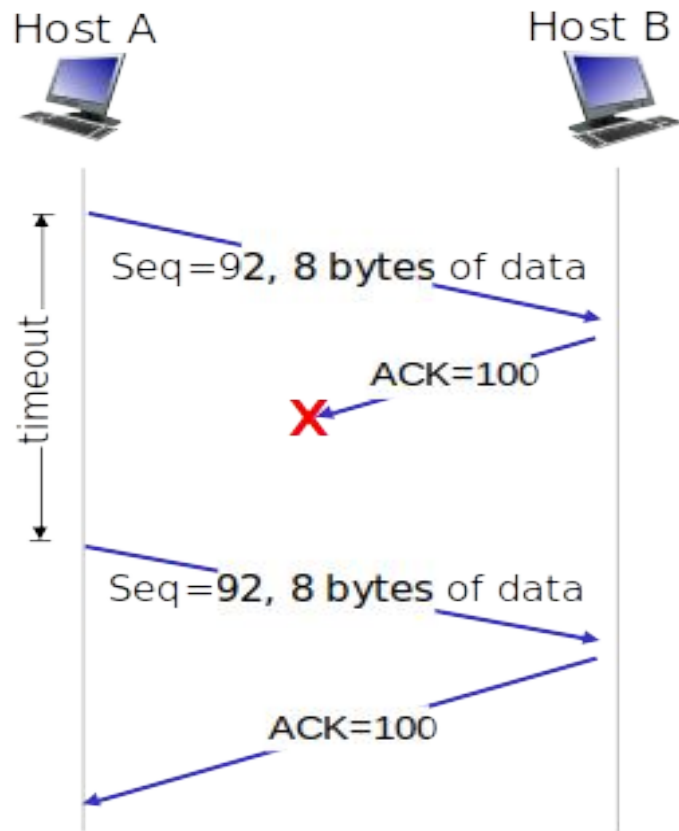


TCP Receiver: ACK generation [RFC 5681]

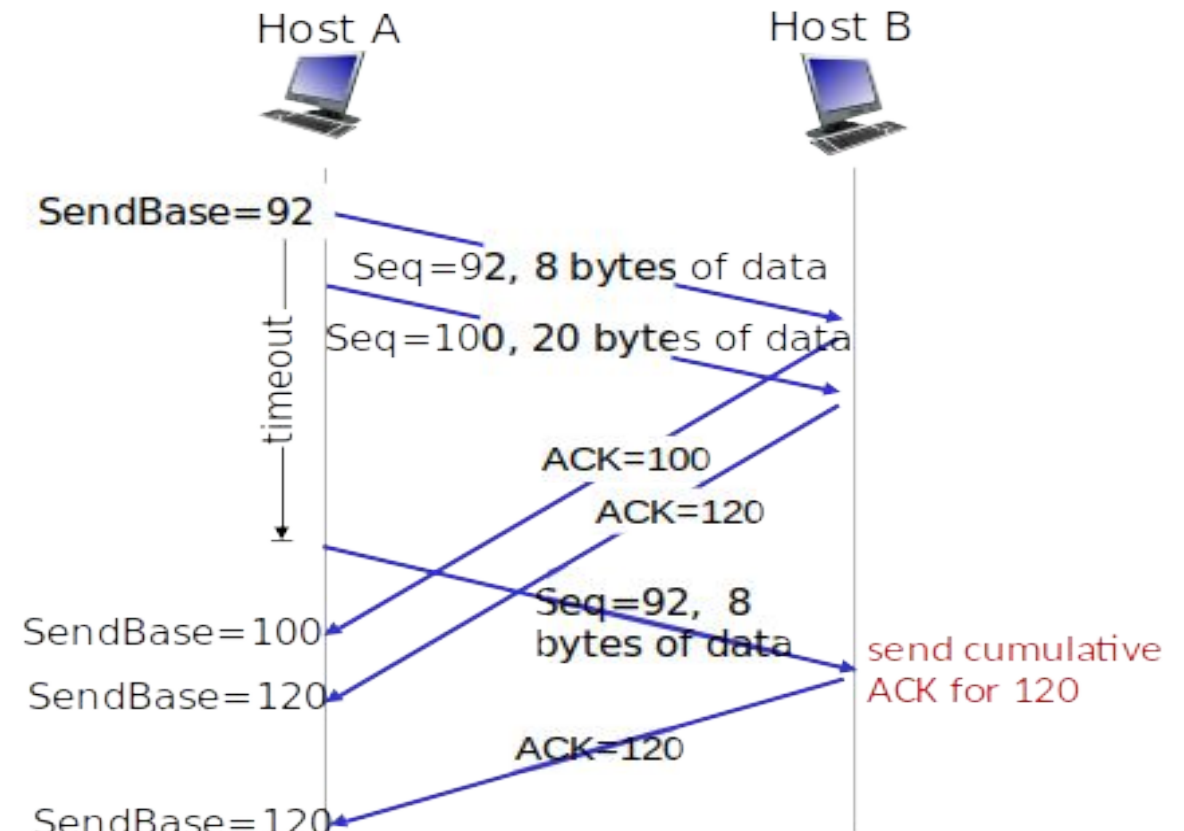


Transport Layer: 3-<number>

TCP: retransmission scenarios



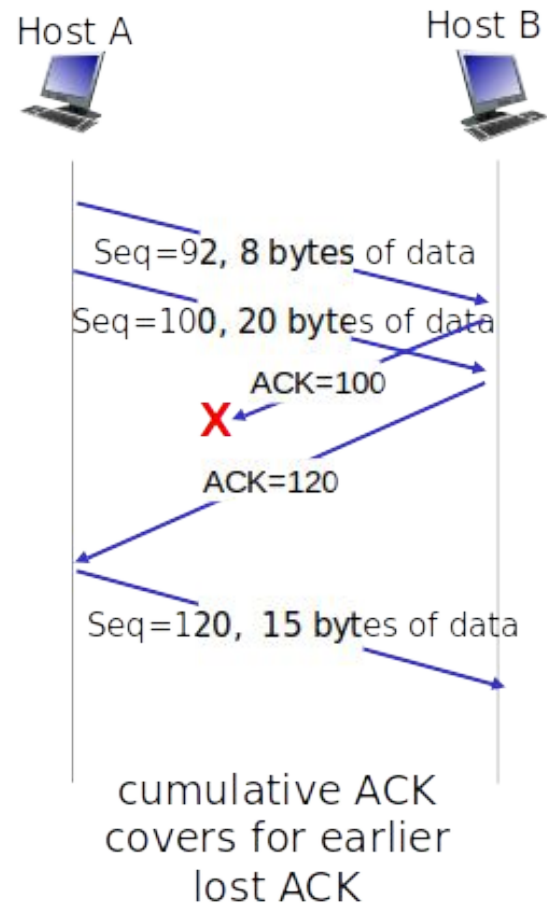
lost ACK scenario



premature timeout

Transport Layer: 3-<number>

TCP: retransmission scenarios



Transport Layer: 3-<number>

TCP fast retransmit

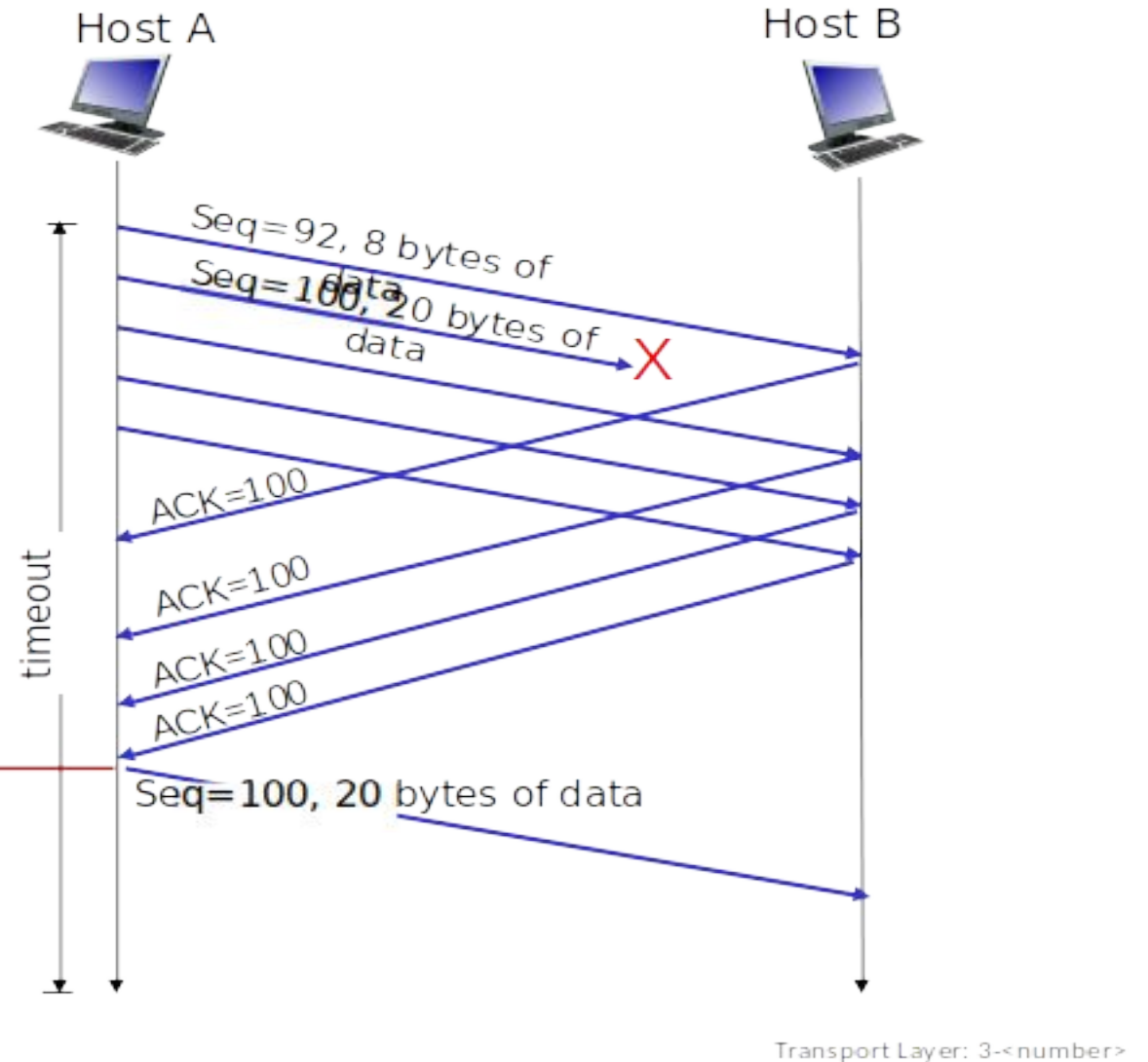
TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost, so don't wait for timeout



Receipt of three duplicate ACKs indicates 3 segments received after a missing segment – lost segment is likely. So retransmit!



Principles of congestion control

Congestion:

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



congestion control:

too many senders,
sending too fast



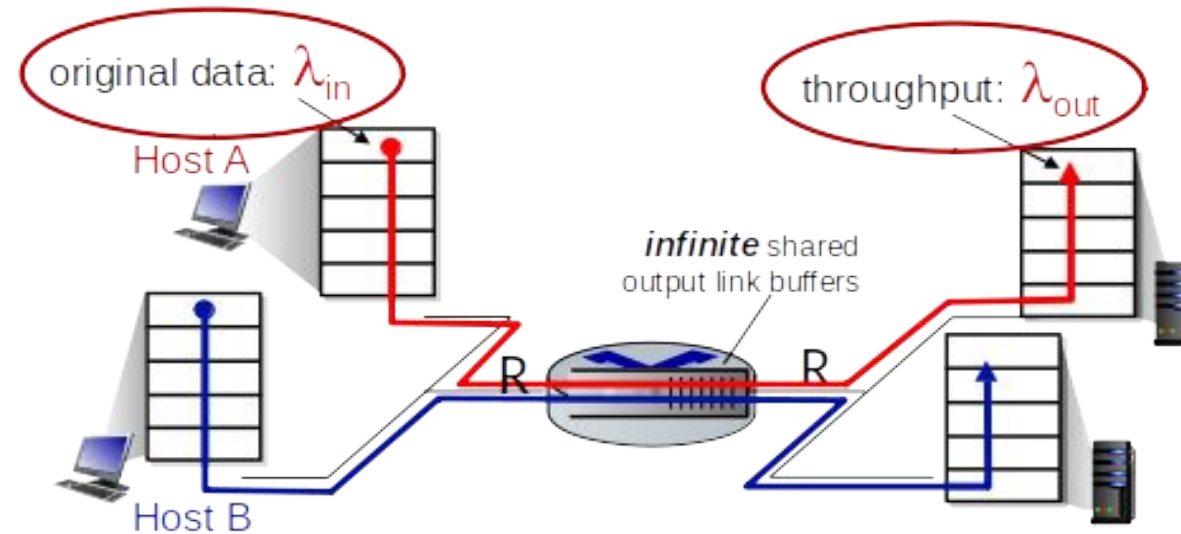
flow control: one sender
too fast for one receiver

Transport Layer: 3-<number>

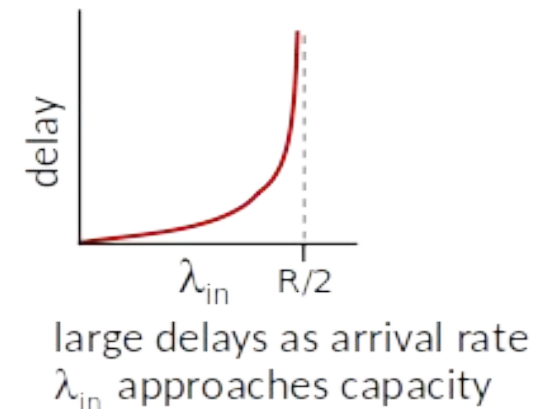
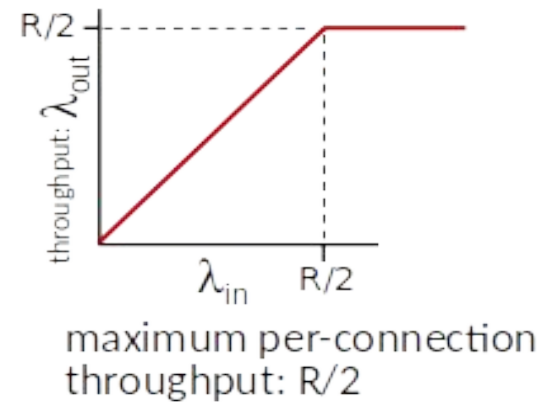
Causes/costs of congestion: scenario 1

Simplest scenario:

- one router, infinite buffers
- input, output link capacity: R
- two flows
- no retransmissions needed



Q: What happens as arrival rate λ_{in} approaches $R/2$?

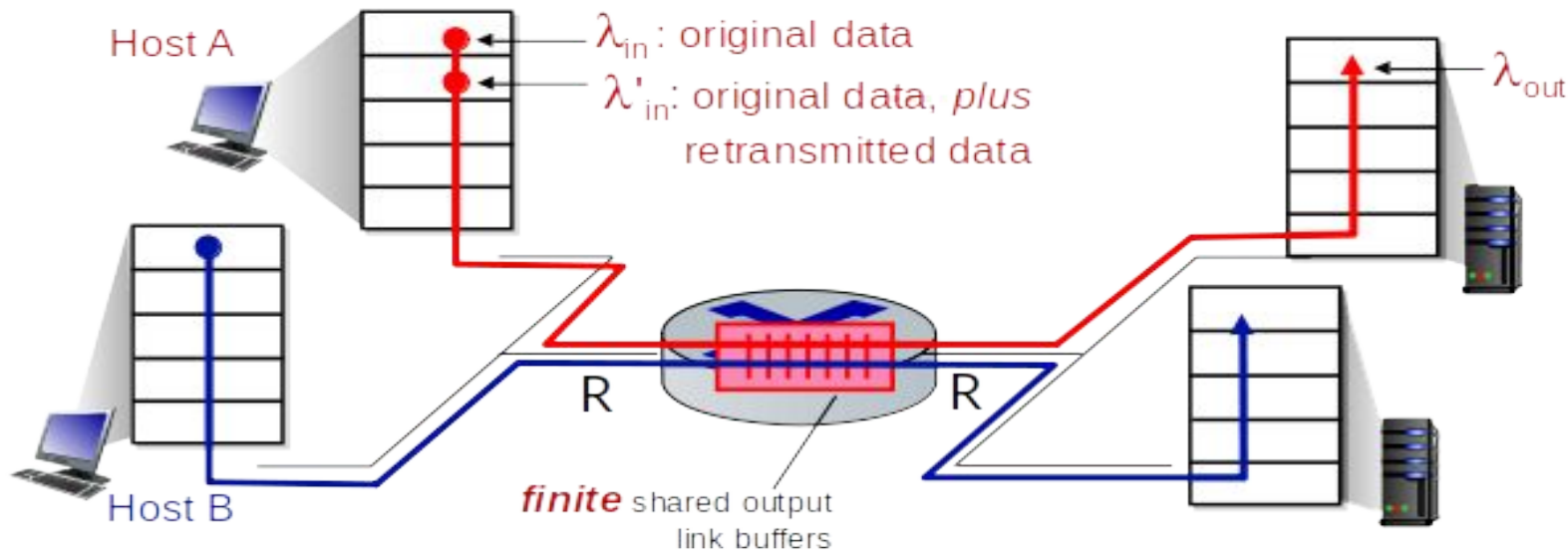


Transport Layer: 3-<number>

R/2

Causes/costs of congestion: scenario 2

- one router, *finite* buffers
- sender retransmits lost, timed-out packet
 - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
 - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$

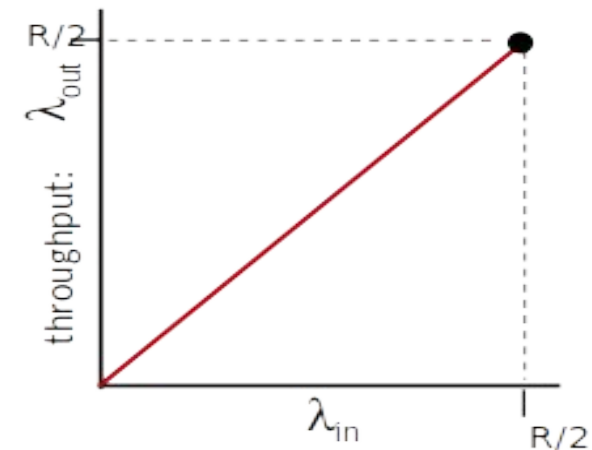
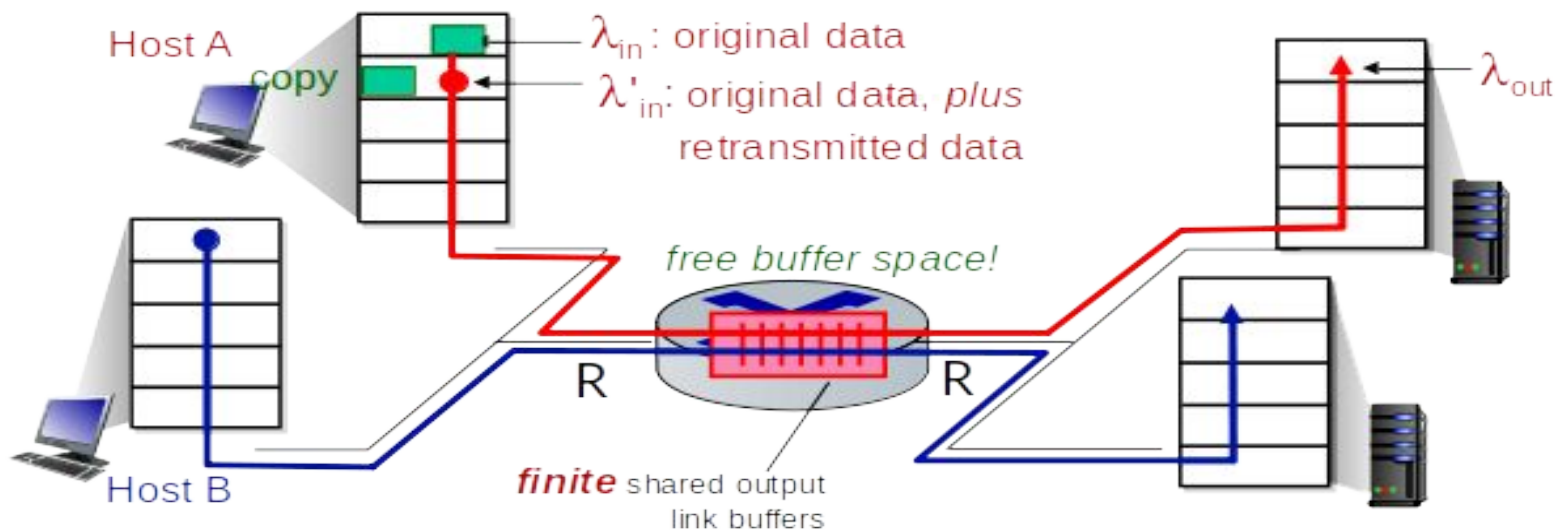


Transport Layer: 3-<number>

Causes/costs of congestion: scenario 2

Idealization: perfect knowledge

- sender sends only when router buffers available

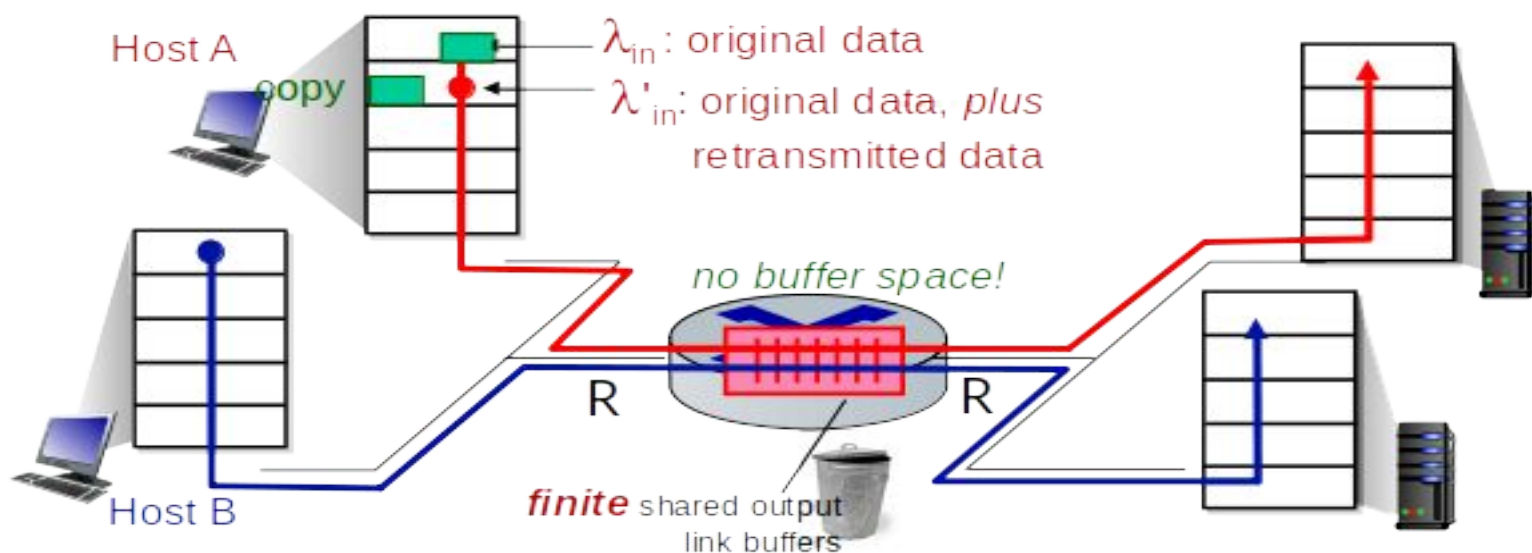


Transport Layer: 3-<number>

Causes/costs of congestion: scenario 2

Idealization: *some* perfect knowledge

- packets can be lost (dropped at router) due to full buffers
- sender knows when packet has been dropped: only resends if packet *known* to be lost

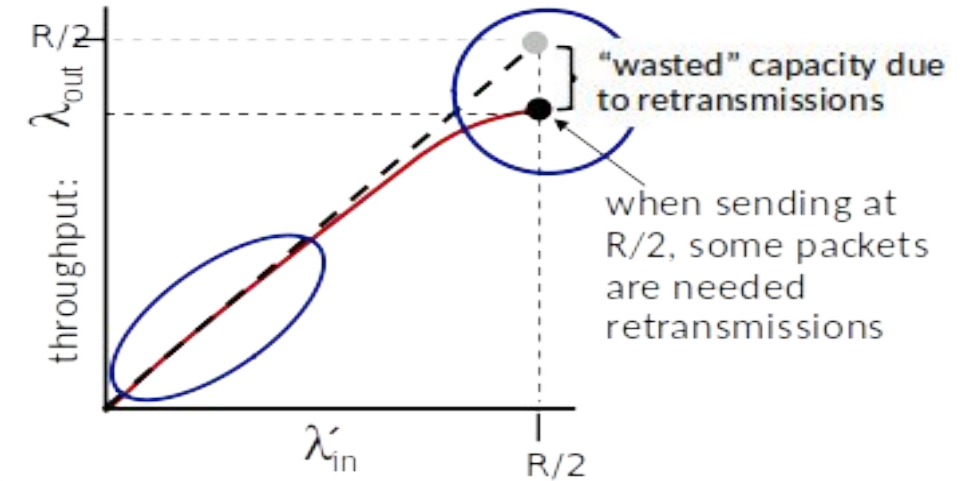
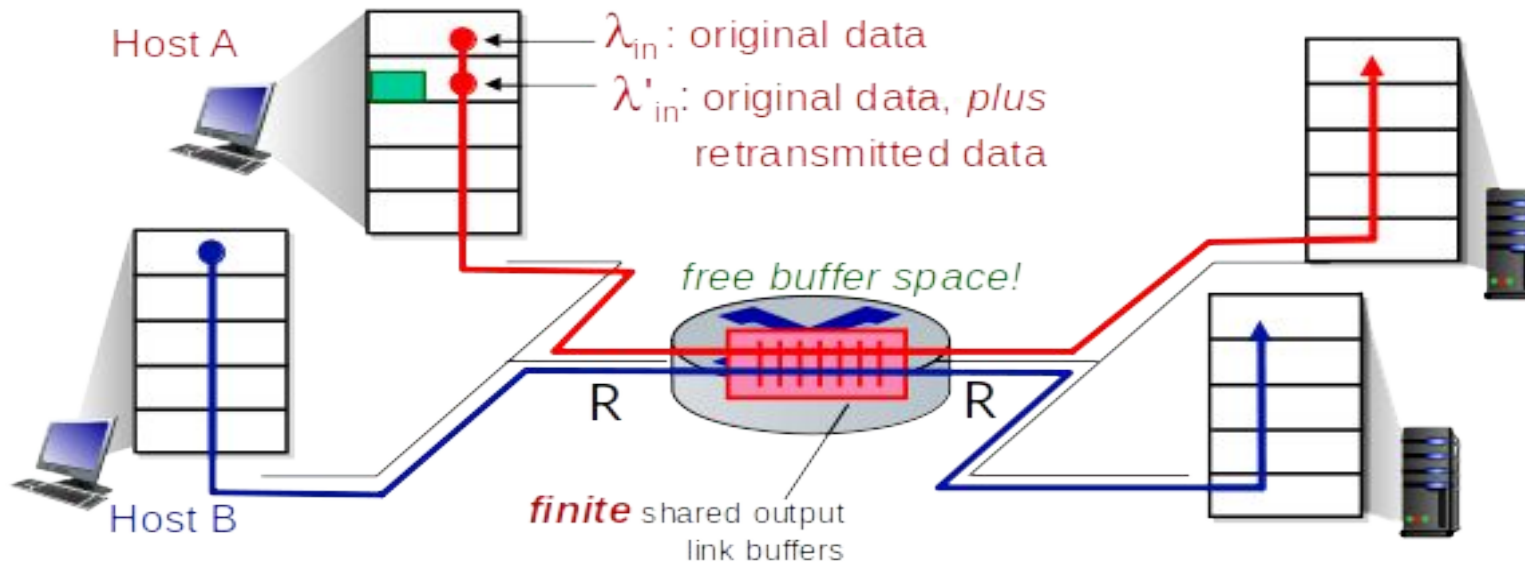


Transport Layer: 3-111

Causes/costs of congestion: scenario 2

Idealization: *some* perfect knowledge

- packets can be lost (dropped at router) due to full buffers
- sender knows when packet has been dropped: only resends if packet *known* to be lost

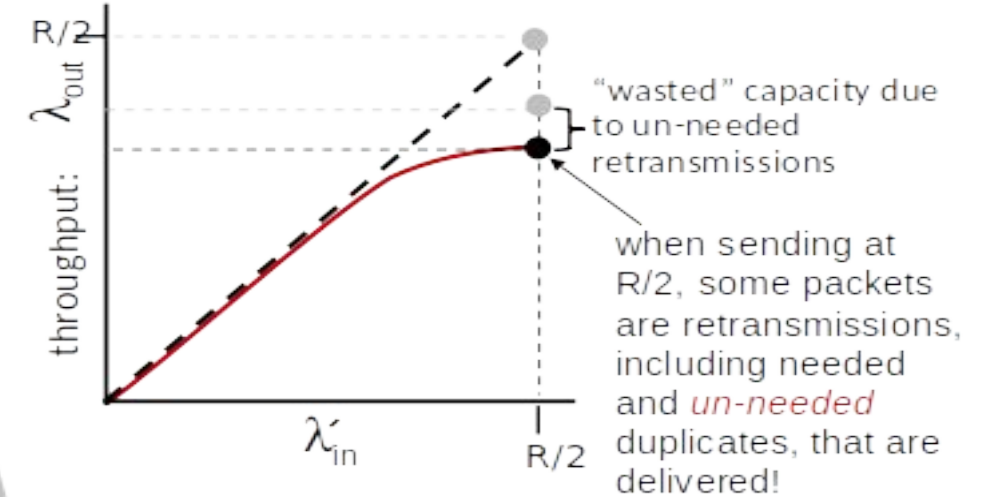
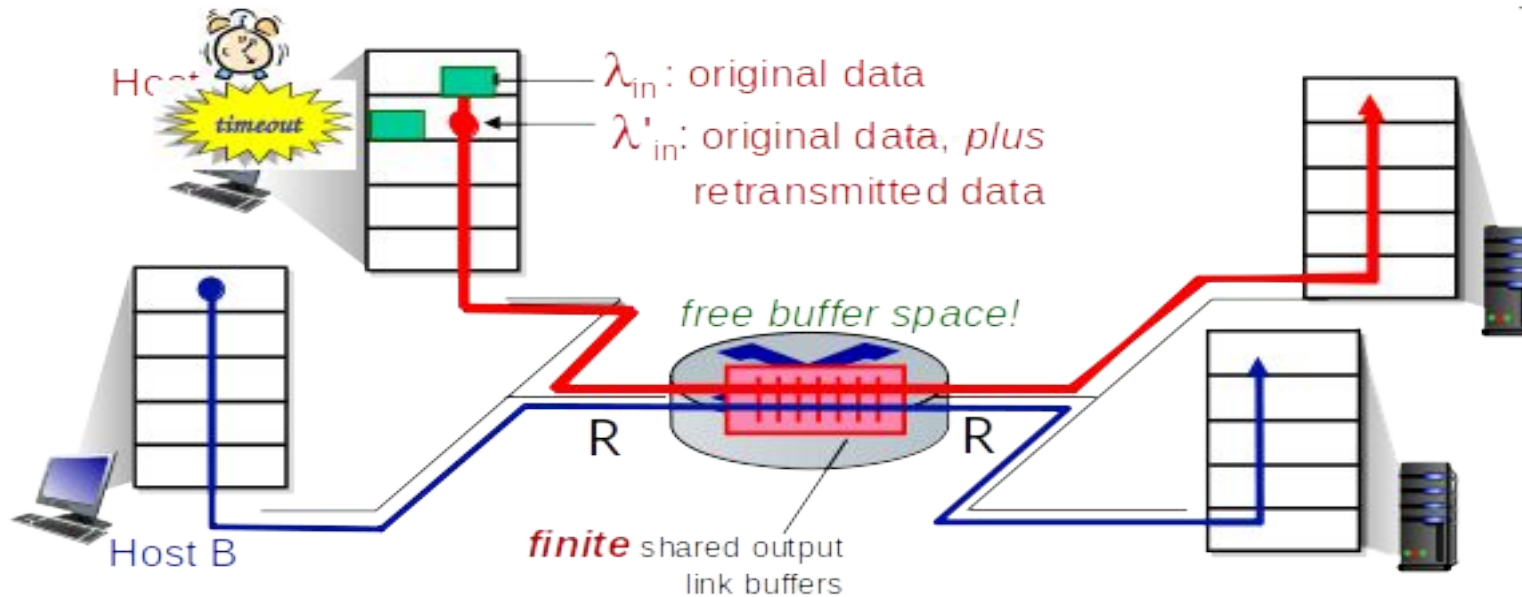


Transport Layer: 3-<number>

Causes/costs of congestion: scenario 2

Realistic scenario: *un-needed duplicates*

- packets can be lost, dropped at router due to full buffers – requiring retransmissions
- but sender times can time out prematurely, sending *two* copies, *both* of which are delivered

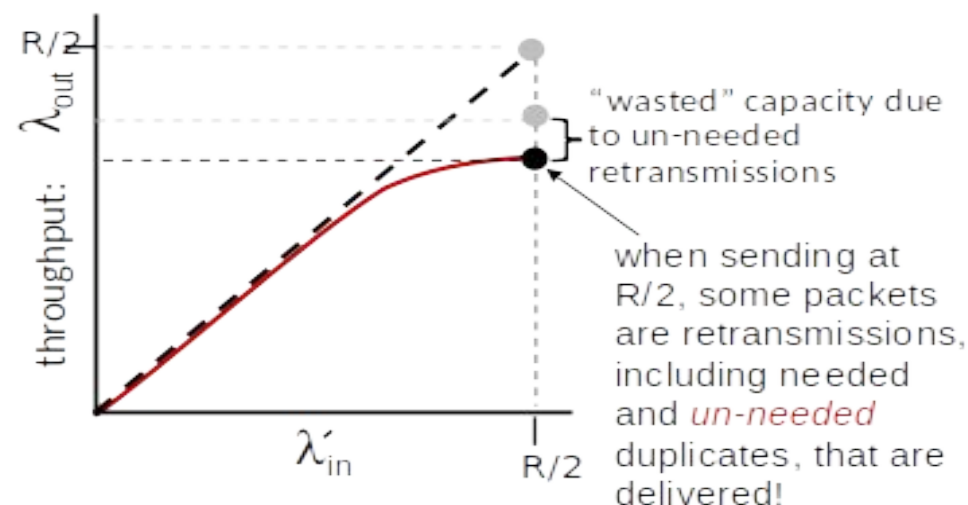


Transport Layer: 3-<number>

Causes/costs of congestion: scenario 2

Realistic scenario: *un-needed duplicates*

- packets can be lost, dropped at router due to full buffers – requiring retransmissions
- but sender times can time out prematurely, sending *two* copies, *both* of which are delivered



"costs" of congestion:

- more work (retransmission) for given receiver throughput
- unneeded retransmissions: link carries multiple copies of a packet
 - decreasing maximum achievable throughput

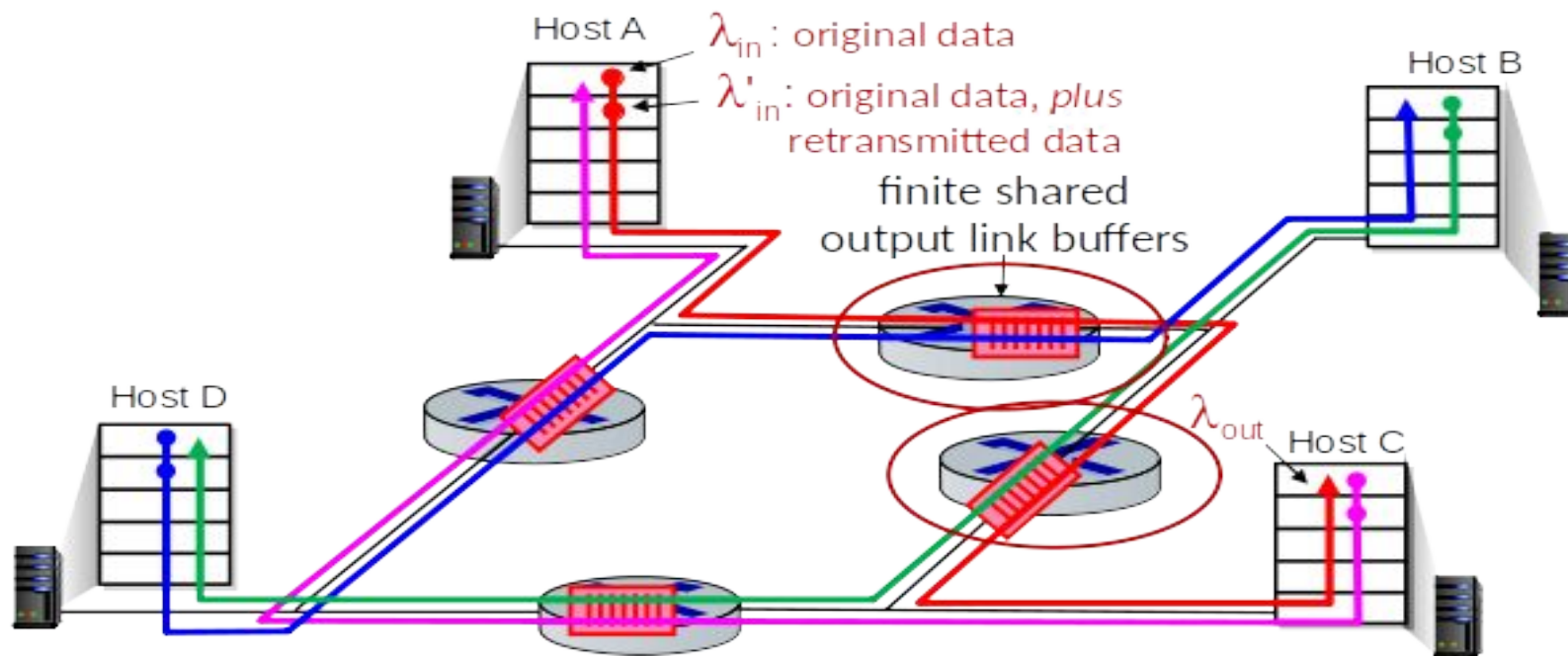
Transport Layer: 3-<number>

Causes/costs of congestion: scenario 3

- four senders
- multi-hop paths
- timeout/retransmit

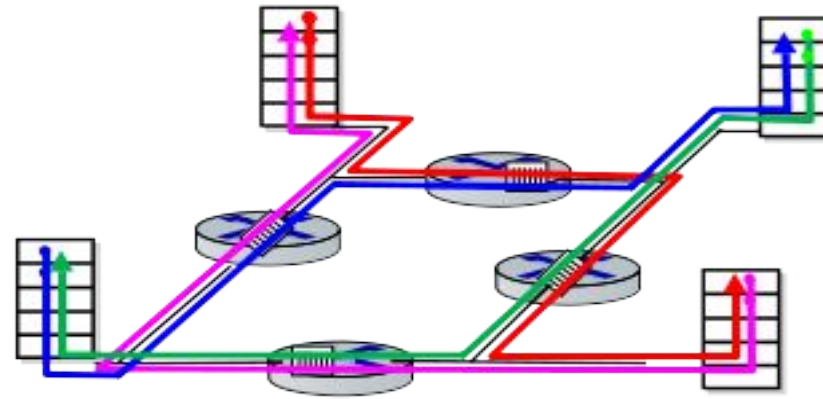
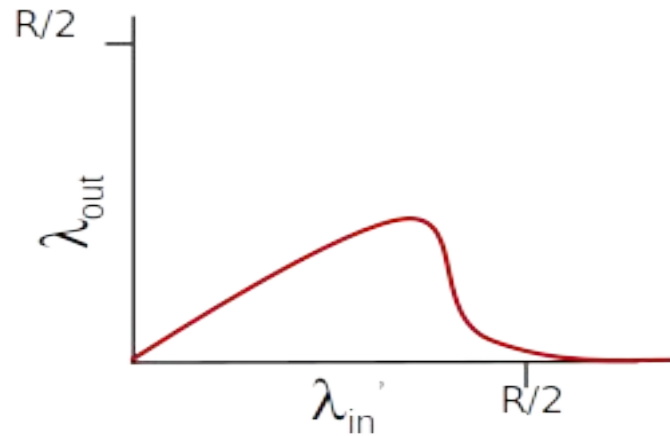
Q: what happens as λ_{in} and λ'_{in} increase ?

A: as red λ'_{in} increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$



Transport Layer: 3-<number>

Causes/costs of congestion: scenario 3



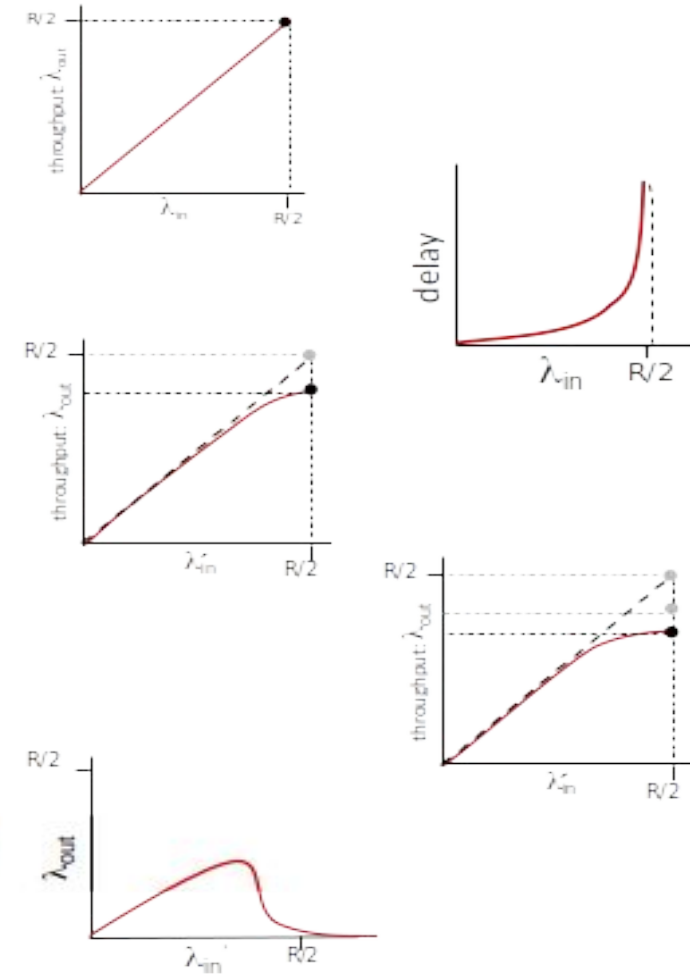
another “cost” of congestion:

- when packet dropped, any upstream transmission capacity and buffering used for that packet was wasted!

Transport Layer: 3-<number>

Causes/costs of congestion: insights

- throughput can never exceed capacity
- delay increases as capacity approached
- loss/retransmission decreases effective throughput
- un-needed duplicates further decreases effective throughput
- upstream transmission capacity / buffering wasted for packets lost downstream

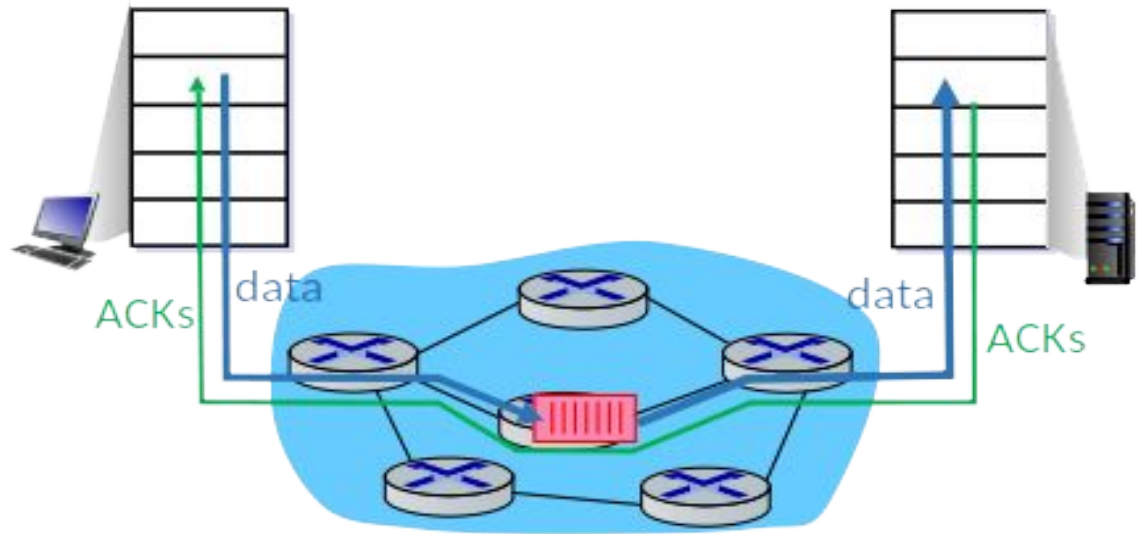


Transport Layer: 3-<number>

Approaches towards congestion control

End-end congestion control:

- no explicit feedback from network
- congestion *inferred* from observed loss, delay
- approach taken by TCP

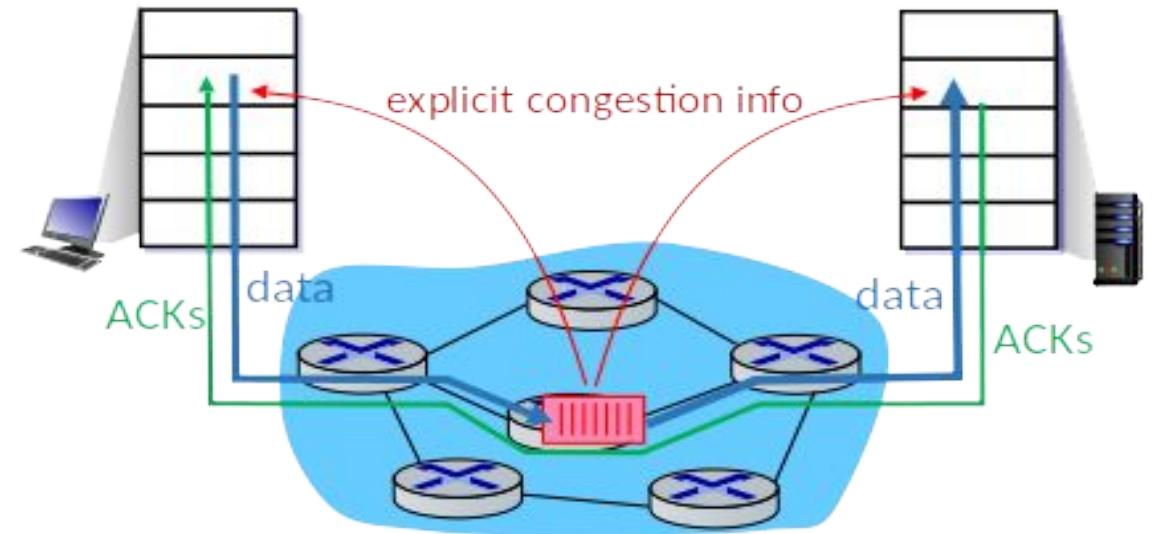


Transport Layer: 3-<number>

Approaches towards congestion control

Network-assisted congestion control:

- routers provide *direct* feedback to sending/receiving hosts with flows passing through congested router
- may indicate congestion level or explicitly set sending rate
- TCP ECN, ATM, DECbit protocols



Transport Layer: 3-119

Thank you