

Problem / Overview

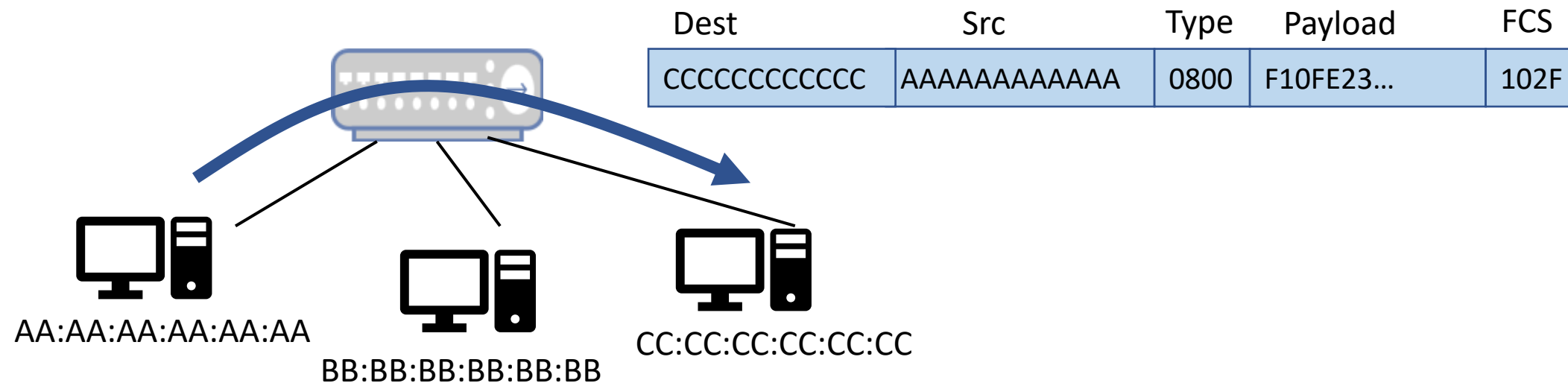
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

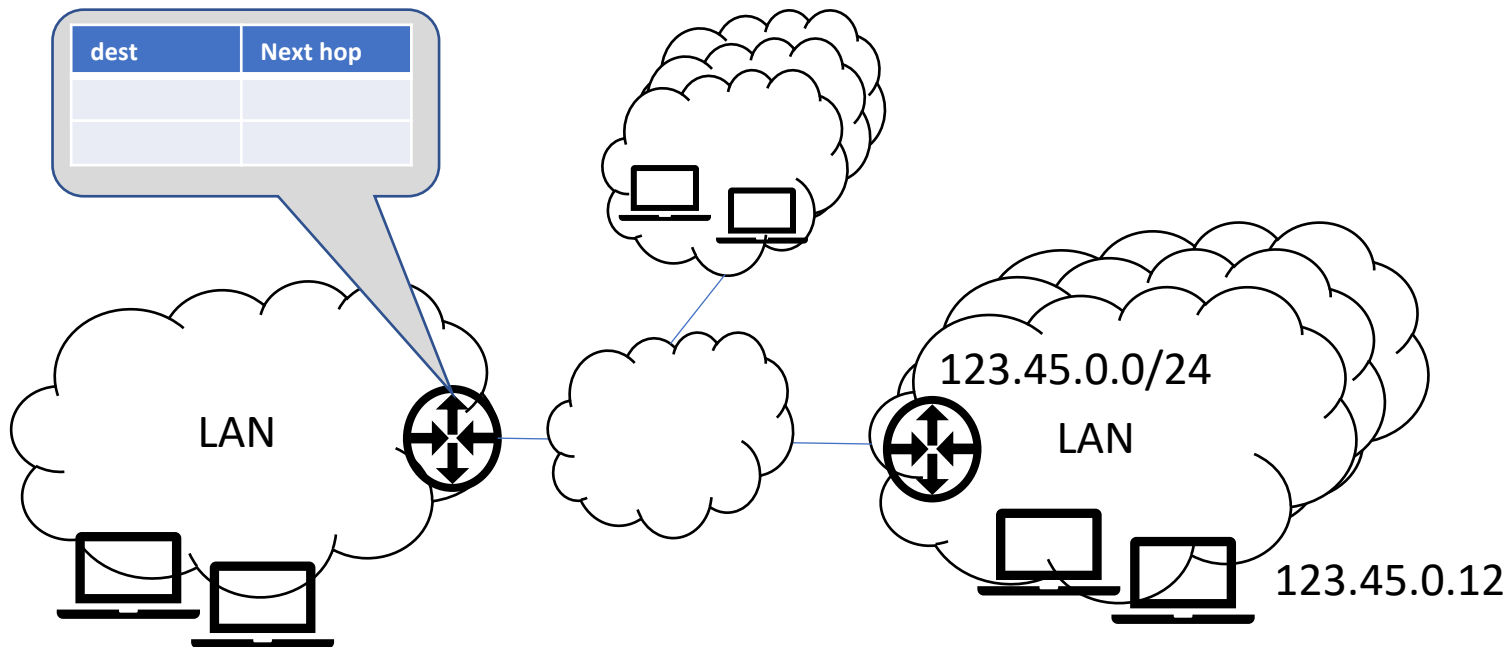
Ethernet Addressing

- Recall:
 - Each NIC has a MAC address
 - Each frame contains src and dest MAC address



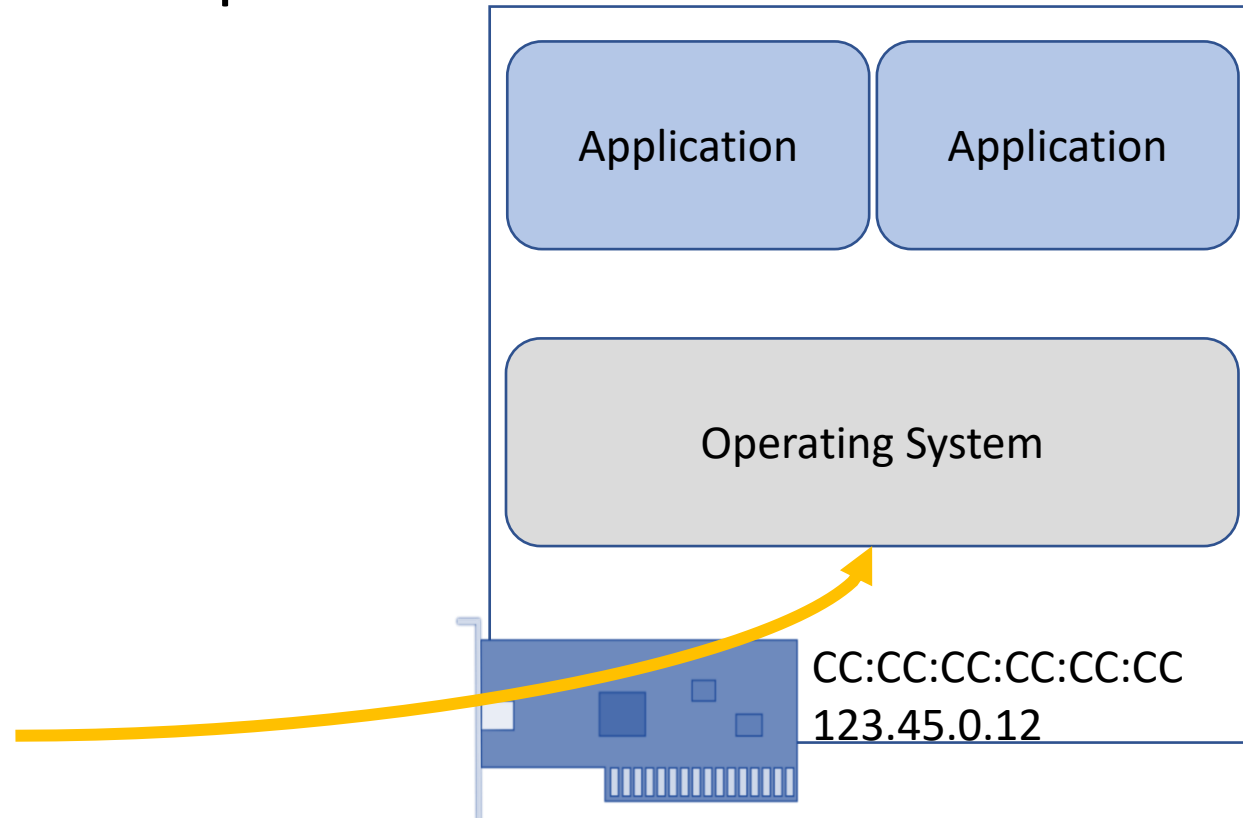
IP Addressing

- In Ethernet we saw table had 1 entry per destination
- That doesn't scale to billions of devices worldwide
- Solution – add structure to addresses so devices can be grouped



Problem 1: Multiplexing

- We have multiple applications running – which application is a packet for?



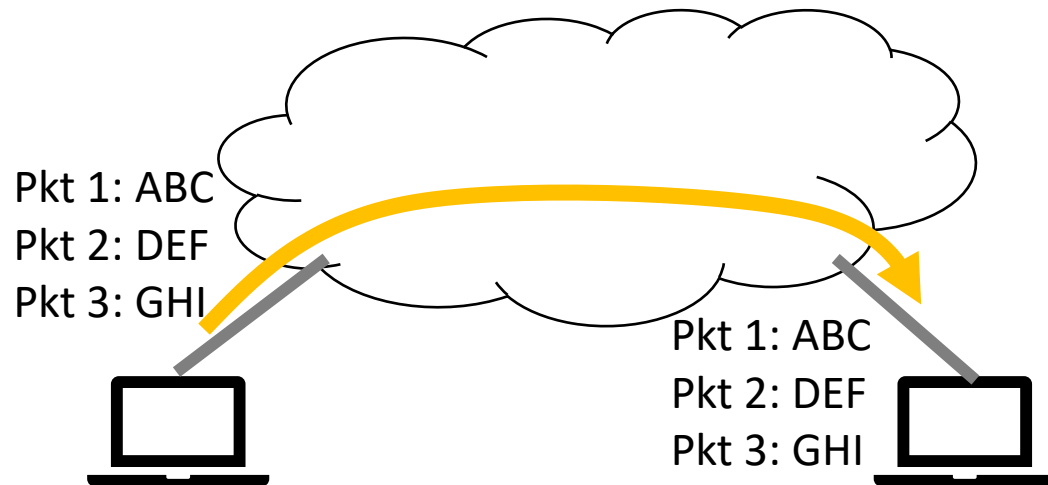
Service Model of the Internet Protocol

- Best Effort
 - Delivery – packets may get dropped
 - Timing – no guarantee on how long it takes to deliver a packet
 - Order – packets may get re-ordered in the network
- Reasoning: inter-connecting different link layers, so needs to be lowest common denominator



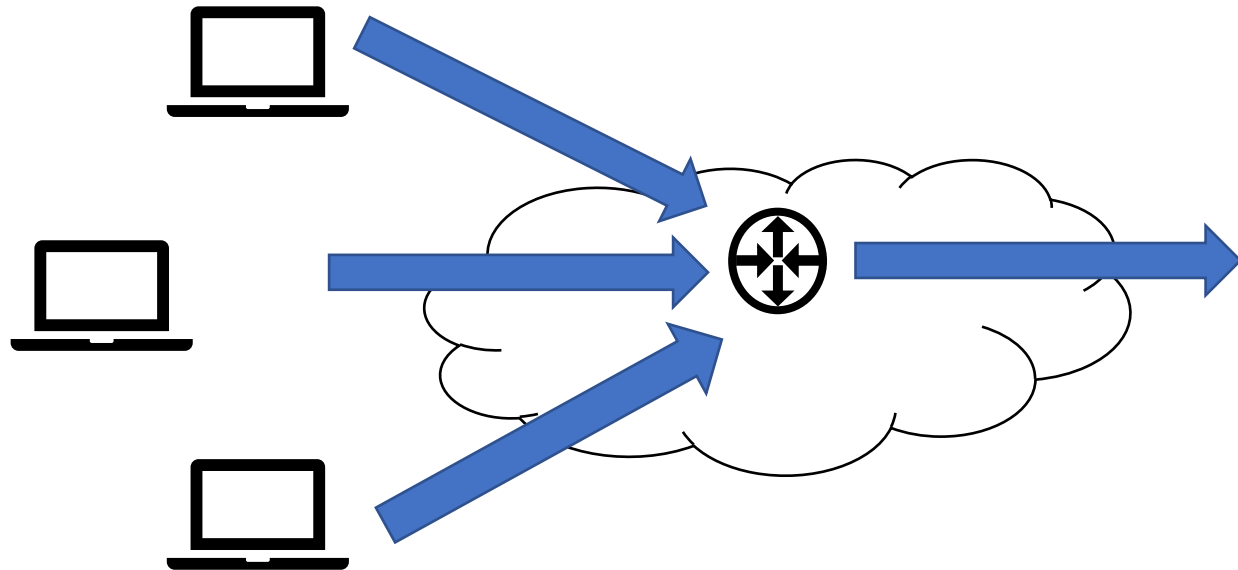
Problem 2: Reliable Transfer

- Goal - Allow an application to be assured that what they send is what is received (same data, same order).



Problem 3: Congestion Control

- Congestion in the network will occur – leading to packets being dropped
- Goal: What if senders could detect this and backoff as needed?



Transport Overview

- Multiplexing
- Reliable Transfer
- Congestion Control





University of Colorado **Boulder**

Multiplexing

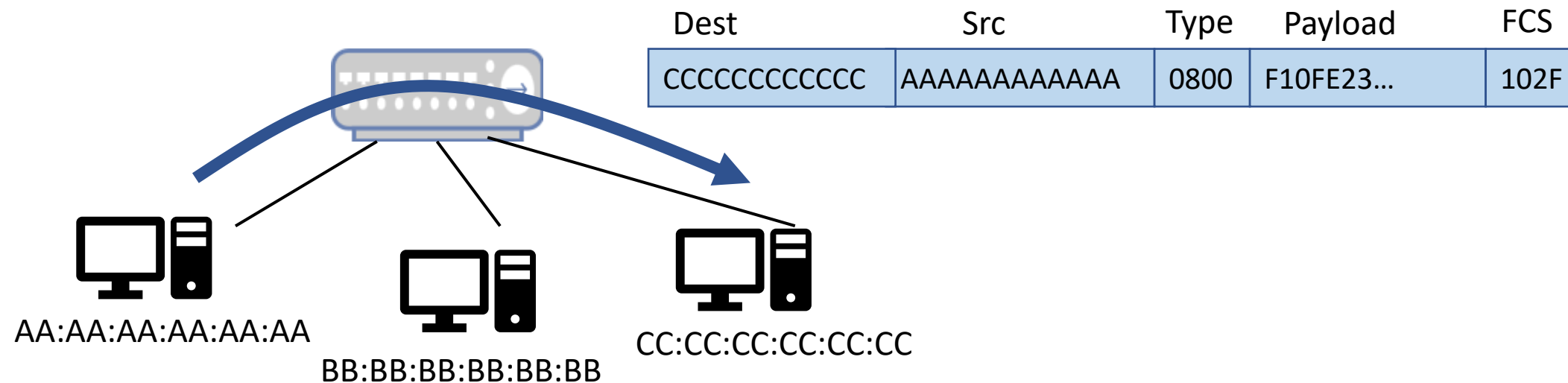
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

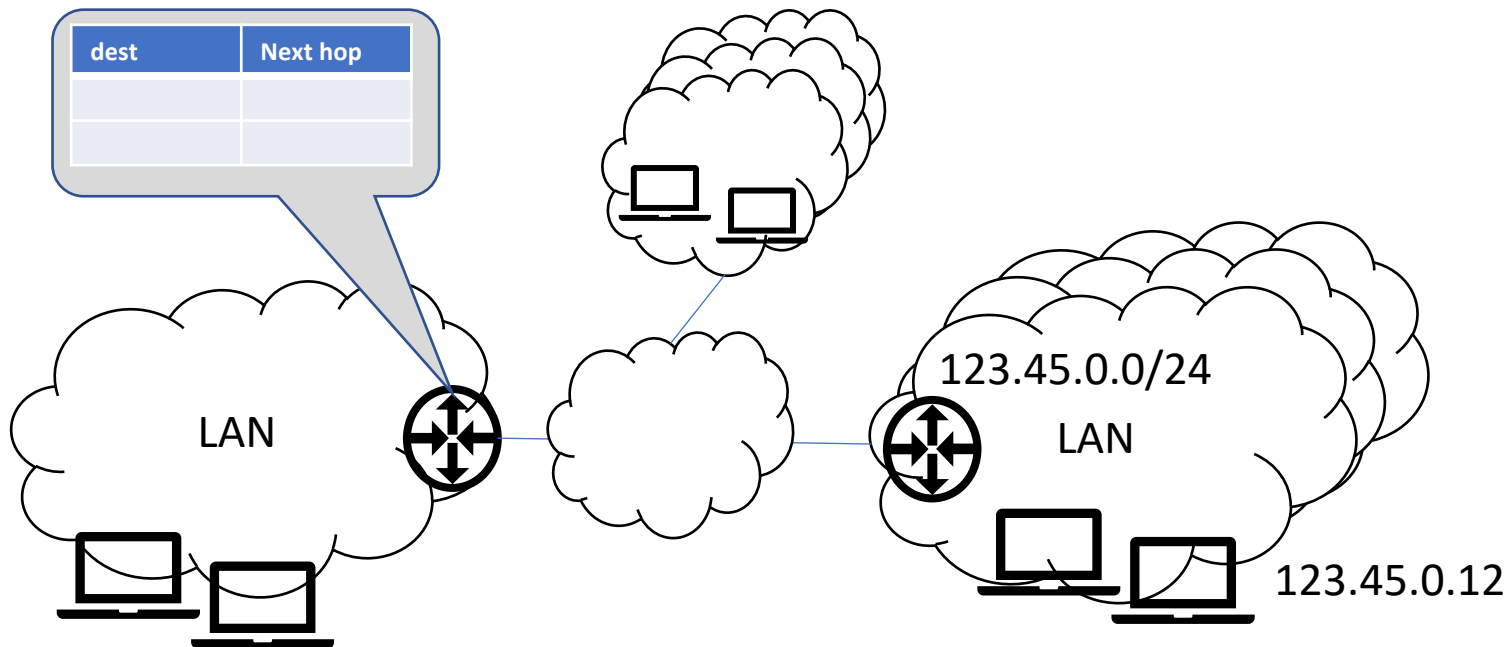
Ethernet Addressing

- Recall:
 - Each NIC has a MAC address
 - Each frame contains src and dest MAC address



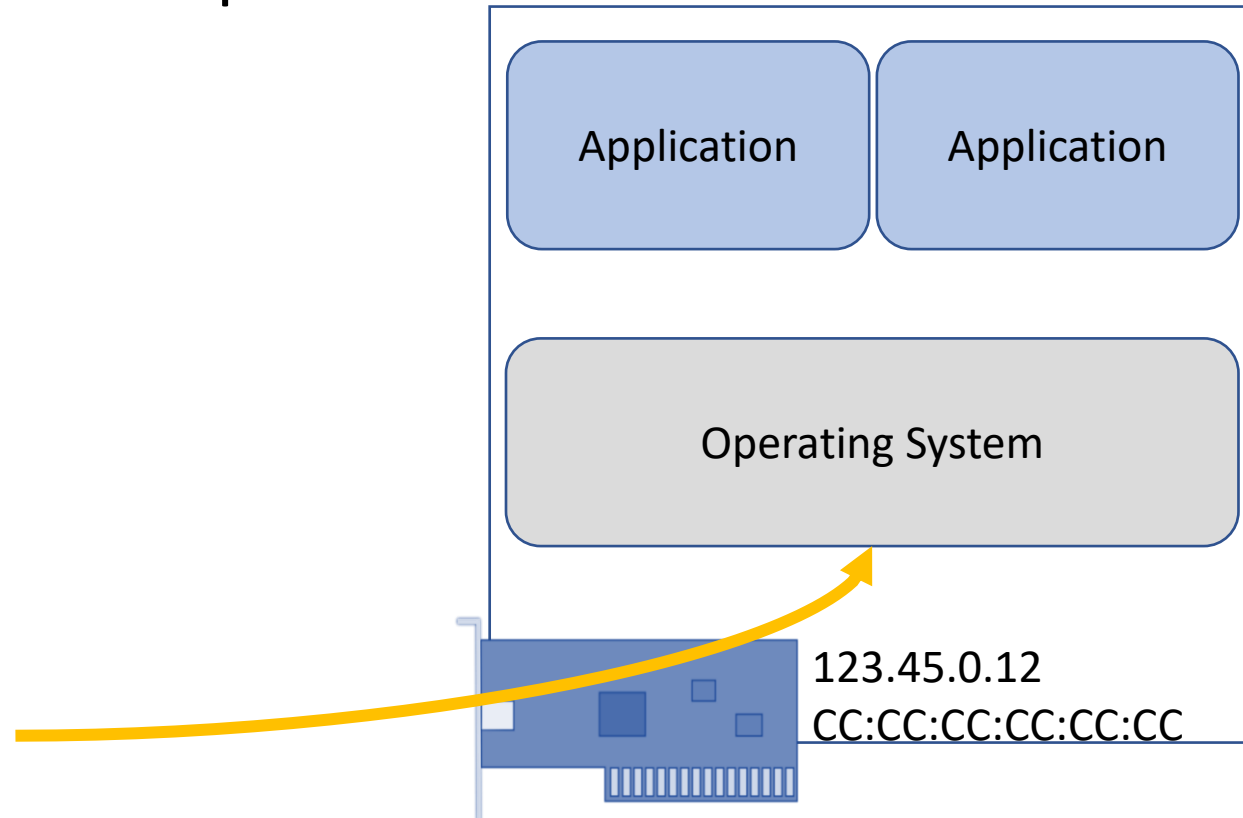
IP Addressing

- In Ethernet we saw table had 1 entry per destination
- That doesn't scale to billions of devices worldwide
- Solution – add structure to addresses so devices can be grouped



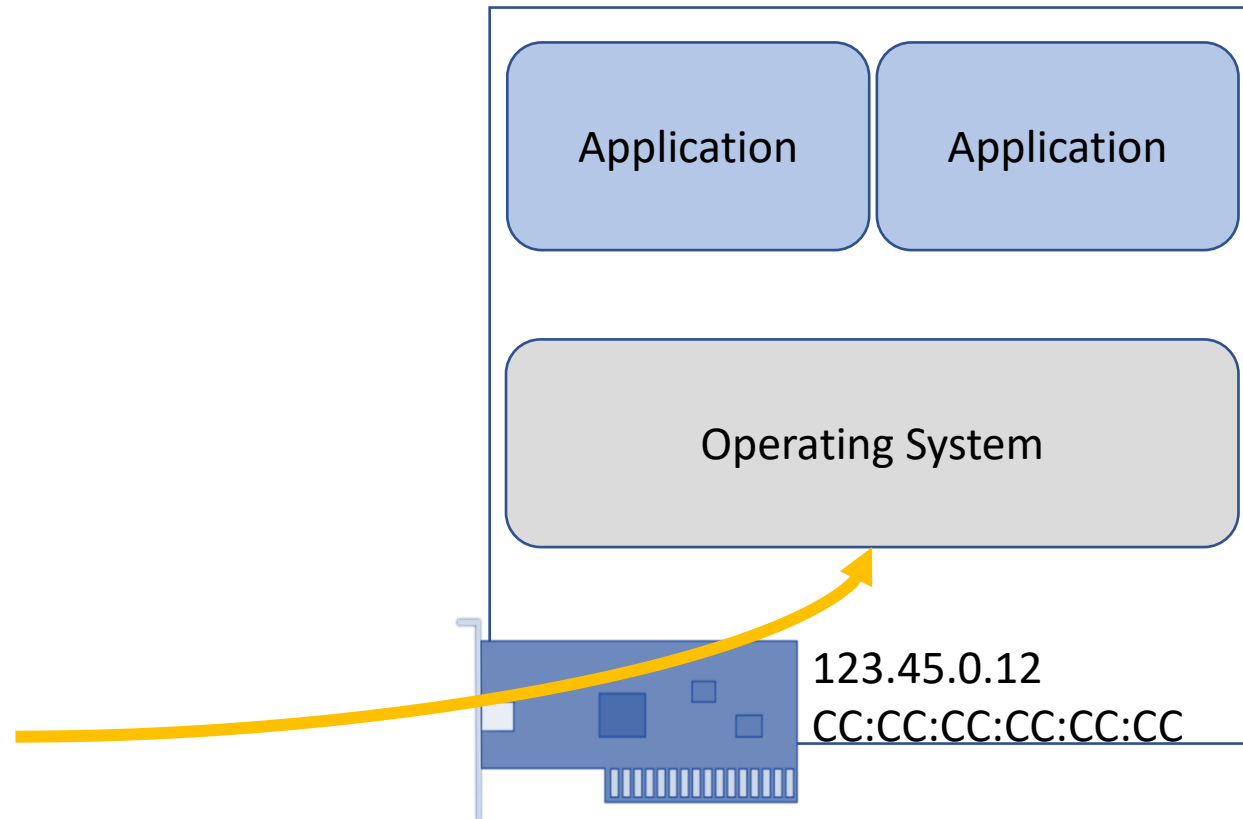
Problem 1: Multiplexing

- We have multiple applications running – which application is a packet for?

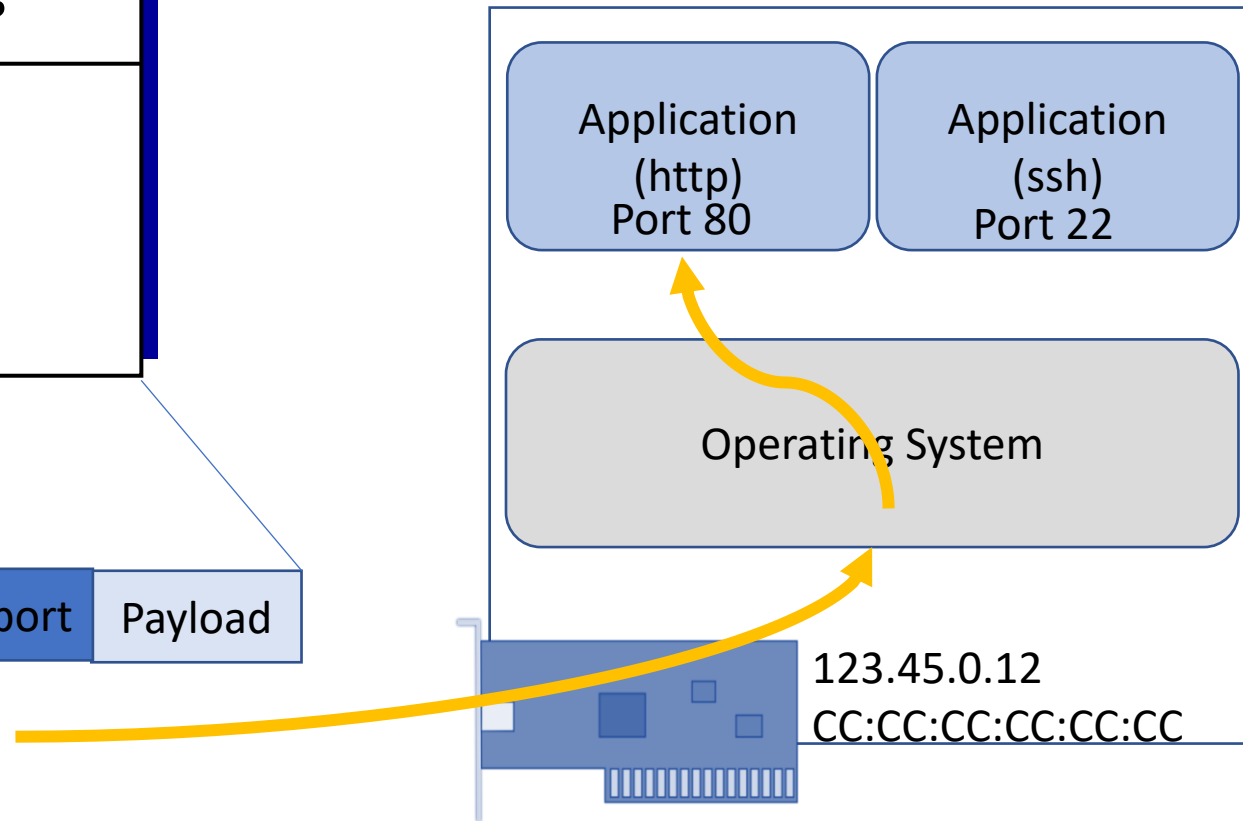
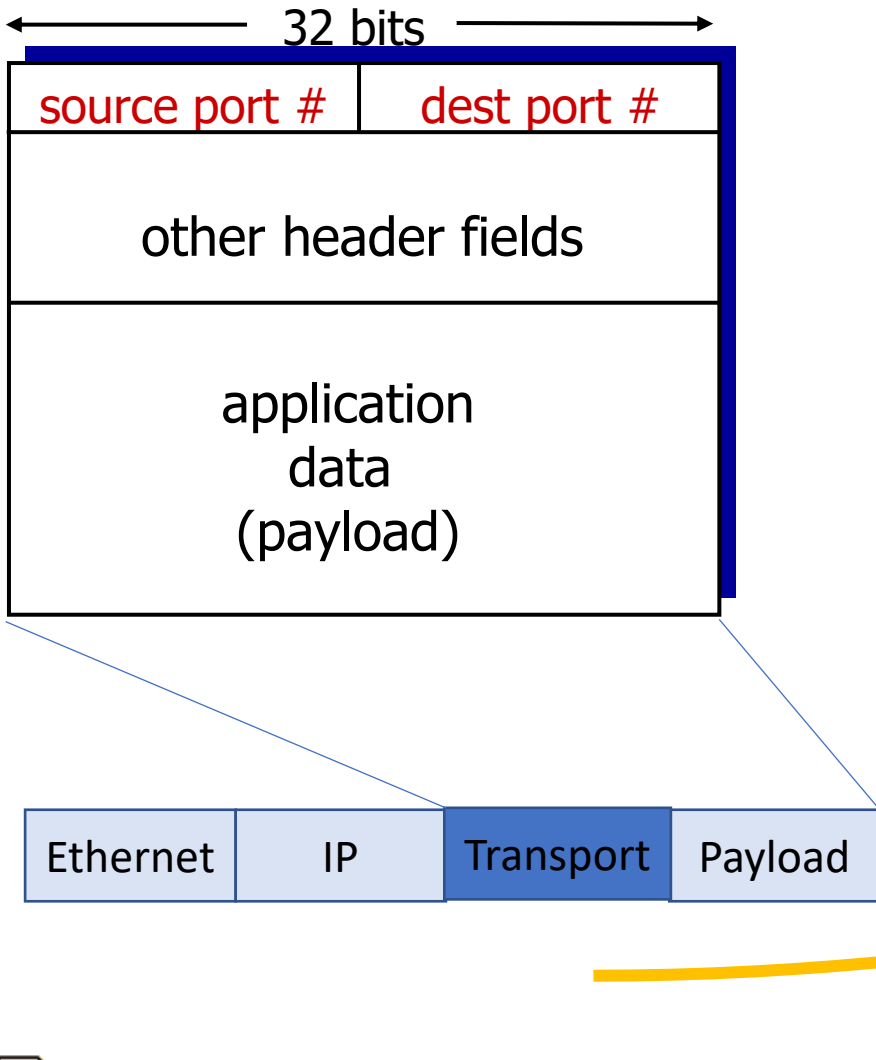


Transport

- Networking layer is for communication between hosts
- Transport layer is for communication between processes



Transport Addressing - Ports

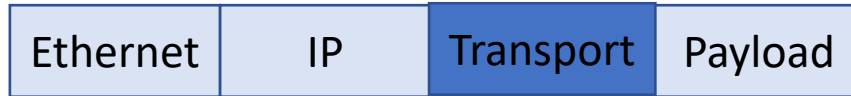


Main Transport Protocols

- UDP
- TCP



Aside – Message Naming

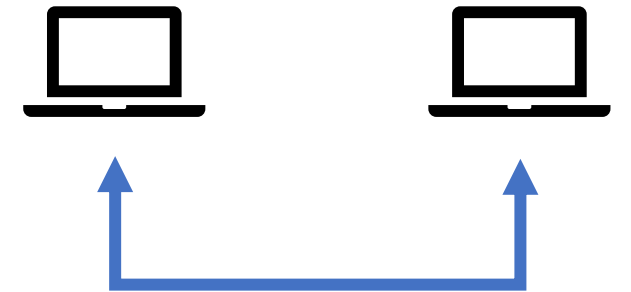


- Link layer – Frame
- Network layer – Packet
- Transport layer – Datagram
- Application layer (next module) - Message



Connectionless vs Connection-oriented

- Connectionless
 - Host uses Dest IP and Dest Port to demultiplex to specific process
 - Doesn't require establishment
 - Only state is (IP, port) → process mapping
- Connection-oriented
 - 4 tuple of Src IP, Dest IP, Src Port, Dest Port identifies the connection
 - Requires establishment before data transfer
 - Host keeps more state about the connection



User Datagram Protocol (UDP)

- Connectionless
- Simple multiplexing / demultiplexing
- Inherits IP's service – best effort delivery of each datagram

UDP datagram header

| Offsets | Octet | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
|---------|-------|-------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Length | | | | | | | | | | | | | | | | Checksum | | | | | | | | | | | | | | | |



Transport Control Protocol (TCP)

- Connection-oriented
- Single process can have multiple connections
 - (src IP, dest IP, src Port, dest Port) identifies the connection
- In-order reliable stream (more in next lessons)

| TCP segment header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|-----|--|---|---|---|---------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0 0 0 0 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



University of Colorado **Boulder**

Reliable Transfer

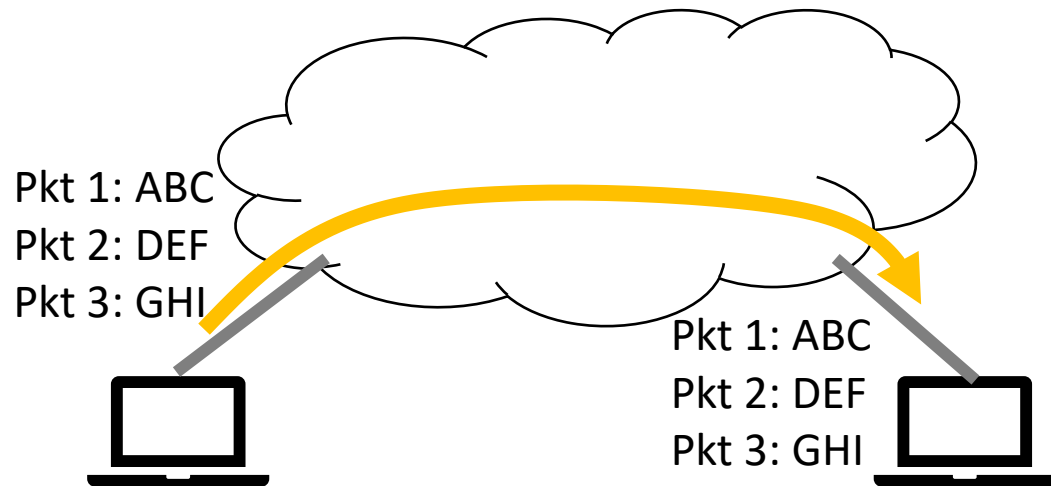
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

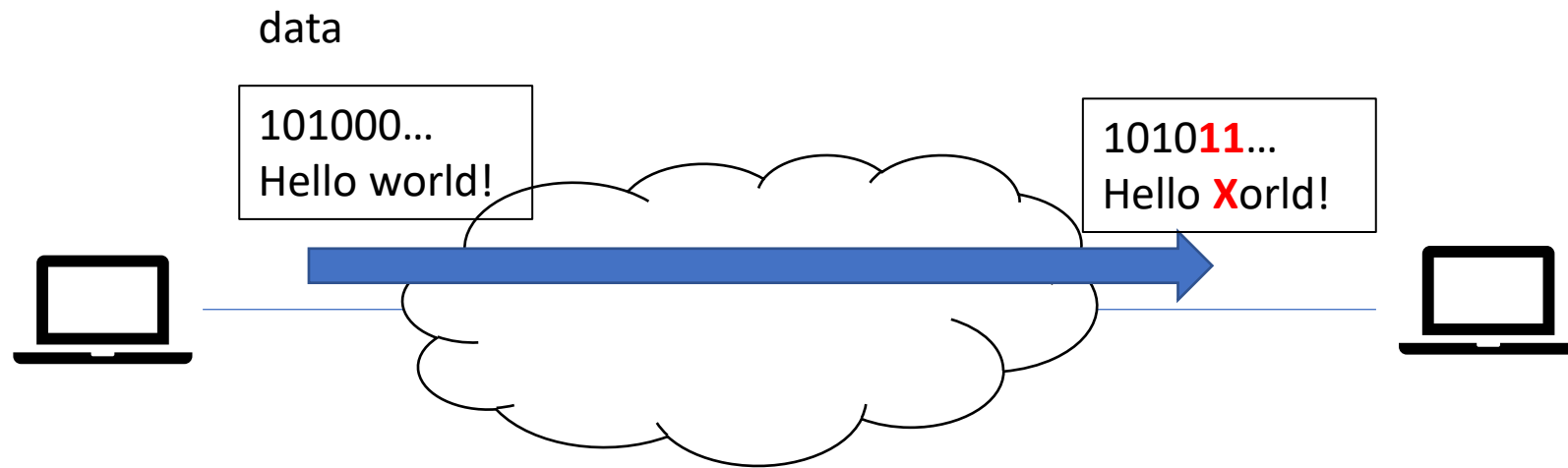
Problem 2: Reliable Transfer

- Goal - Allow an application to be assured that what they send is what is received (same data, same order).



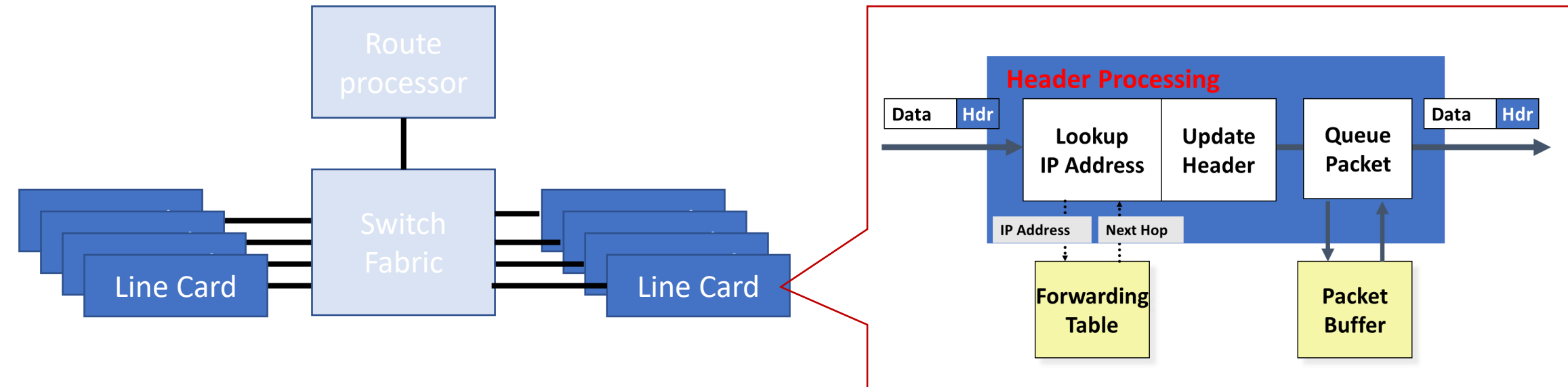
Motivation 1: Frame/Package Error

- Recall – at link layer errors may occur (e.g., from interference)
- Ethernet has a CRC field to be able to detect errors
- But not fix errors – so, drop frames that are in error



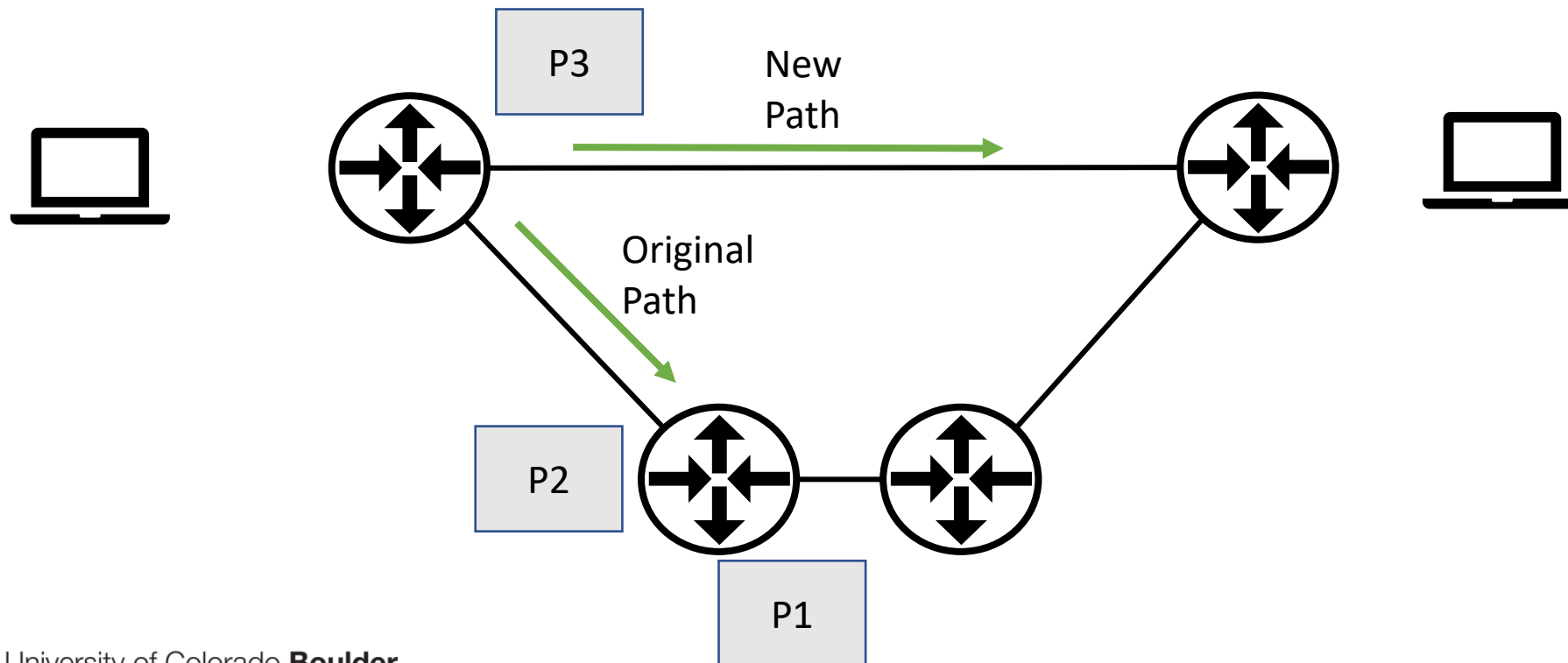
Motivation 2: Dropping due to Congestion

- Recall – router architecture queues packets for transmission.
- If incoming rate $>$ outgoing rate then buffer will fill up



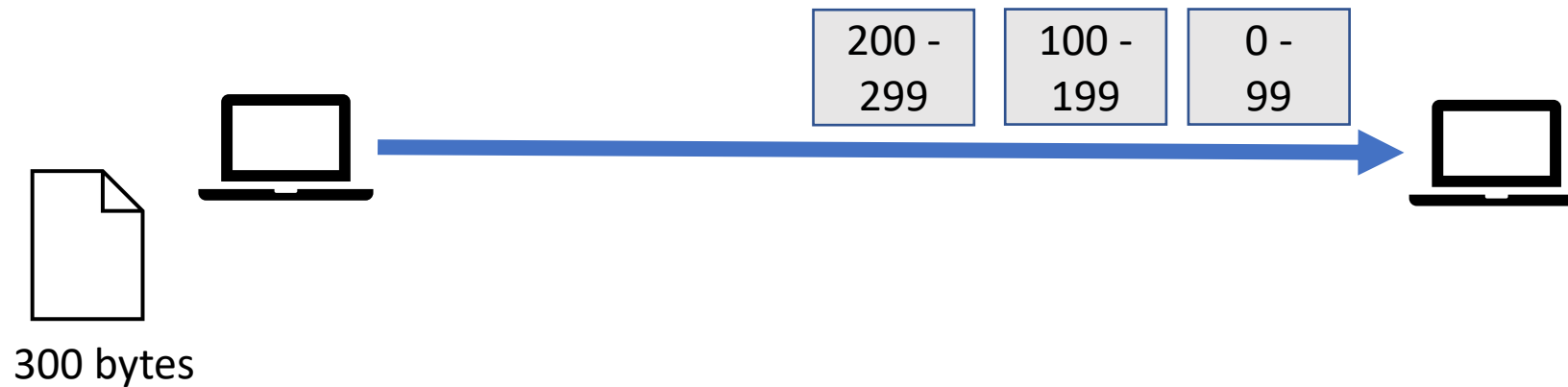
Motivation 3: Packet Re-ordering

- Recall – routing calculates best path, and can change over time
- In example – new path is shorter transmission delay than old



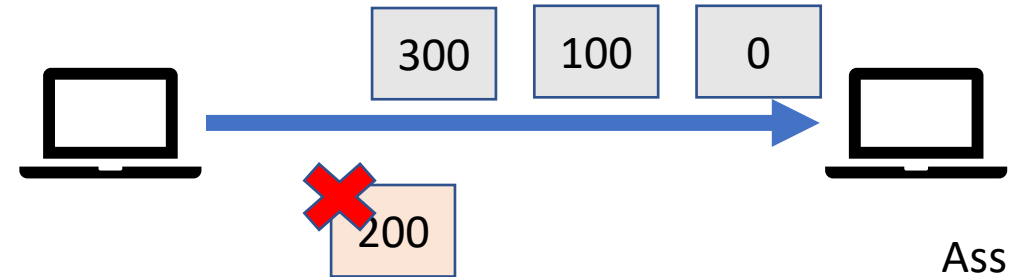
TCP Mechanisms for Reliable Transfer

- Recall – TCP provides abstraction of a reliable in order stream



TCP Mechanisms for Reliable Transfer (1)

- Sequence Number
 - Indicates which bytes in the stream the datagram contains



Assume each is
length 100 bytes

| TCP segment header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|-----|--|---|---|---|------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0000 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

TCP Mechanisms for Reliable Transfer (2)

- Acknowledgement Number
 - Specifies what is the next byte in the stream receiver expects



| TCP segment header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|-----|--|---|---|---|------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0000 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

TCP Mechanisms for Reliable Transfer (3)

- Re-transmission
 - Sender can re-send unacknowledged bytes in the stream



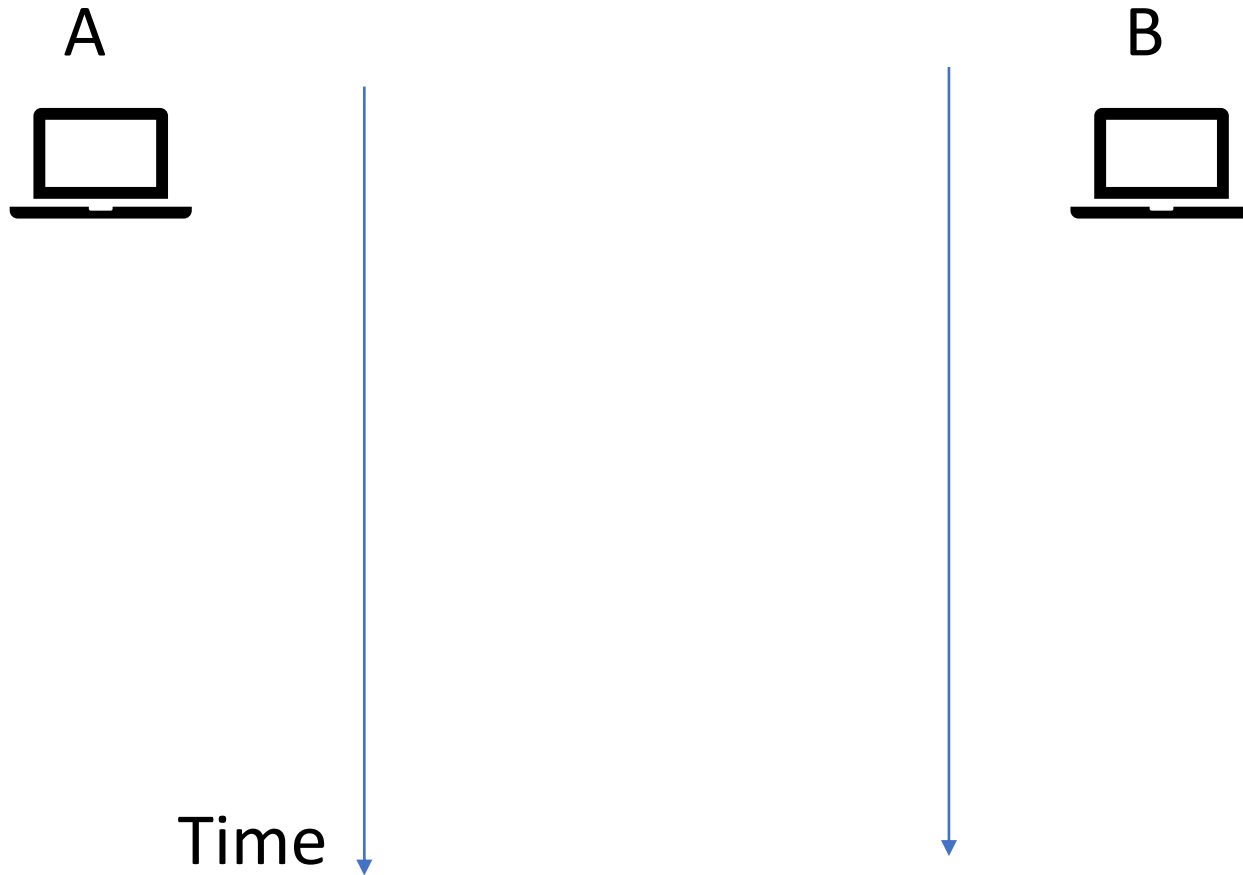
| TCP segment header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|-----|--|---|---|---|------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0000 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

When to Re-transmit

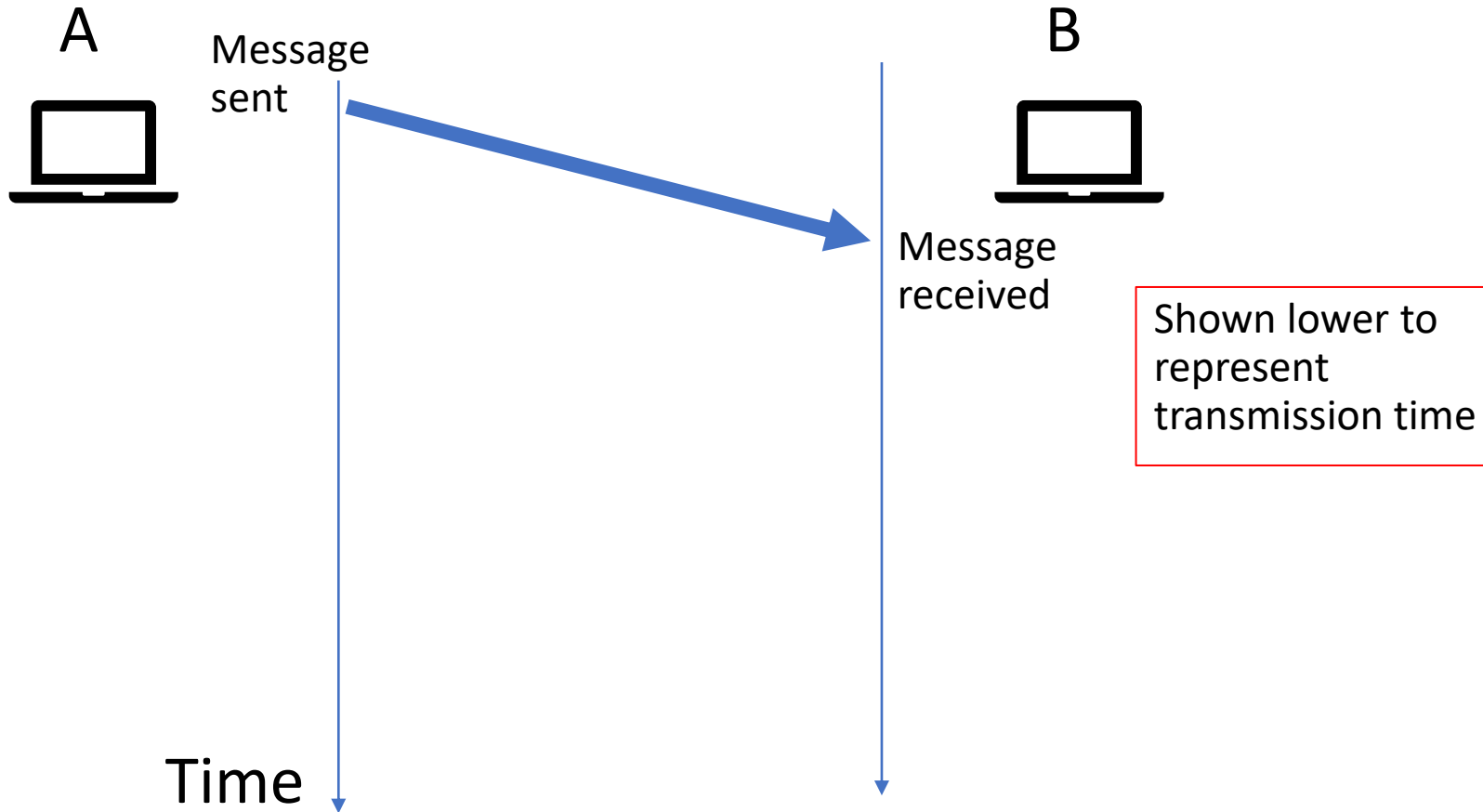
- Timeout
- ACK



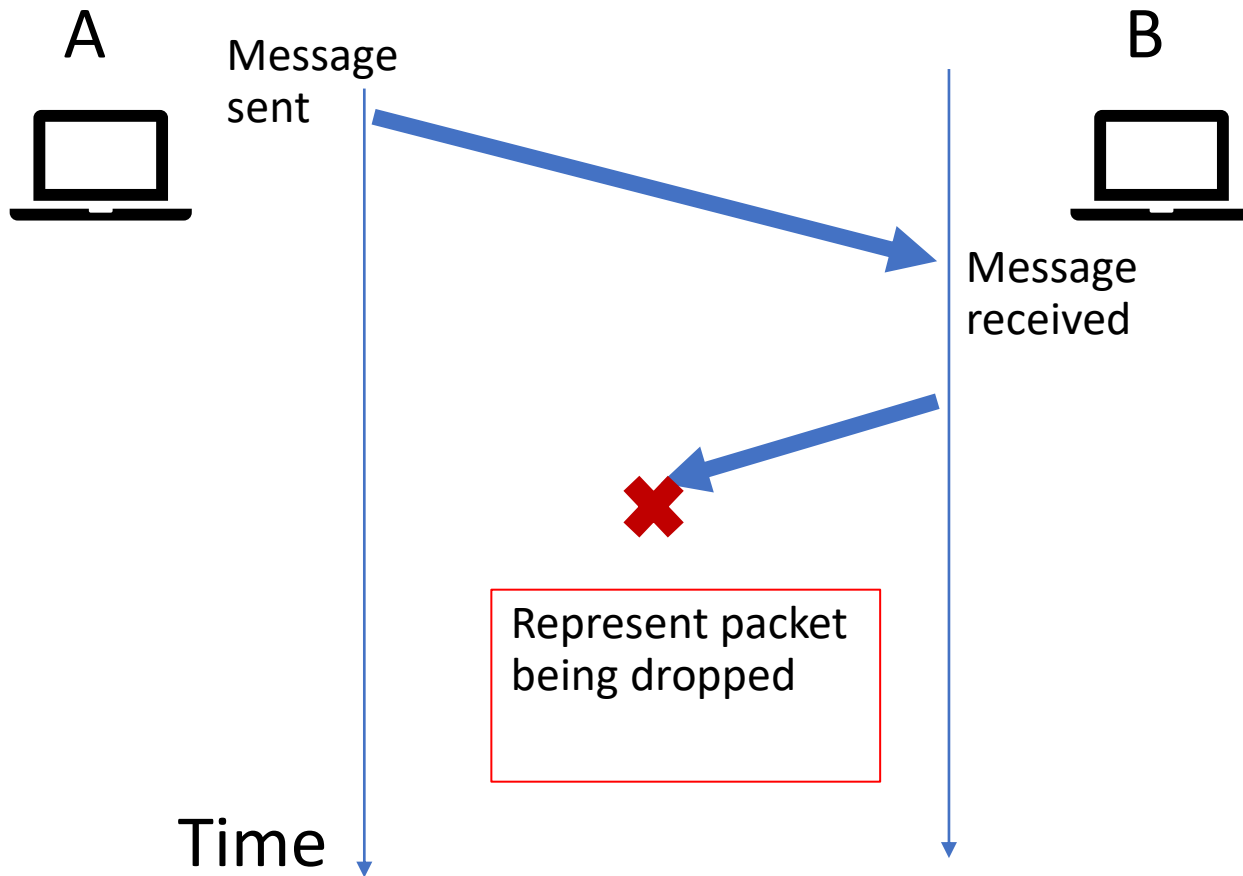
Aside – Message Exchange Diagram



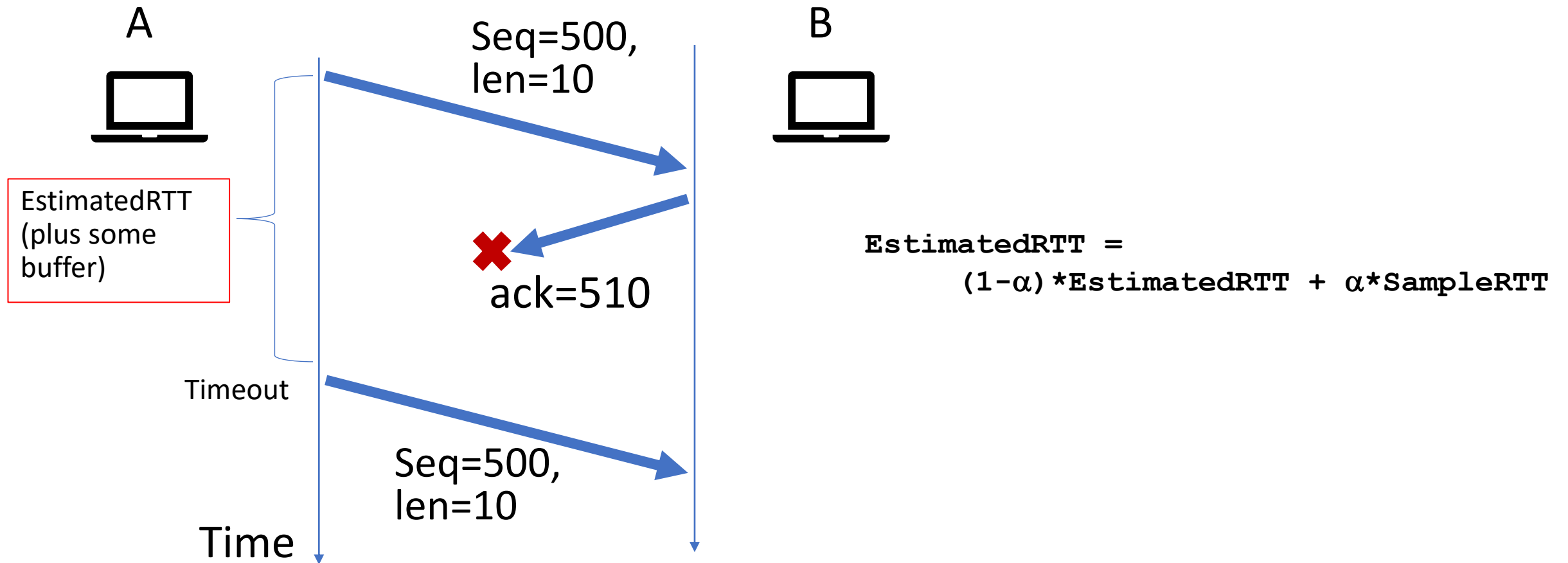
Aside – Message Exchange Diagram



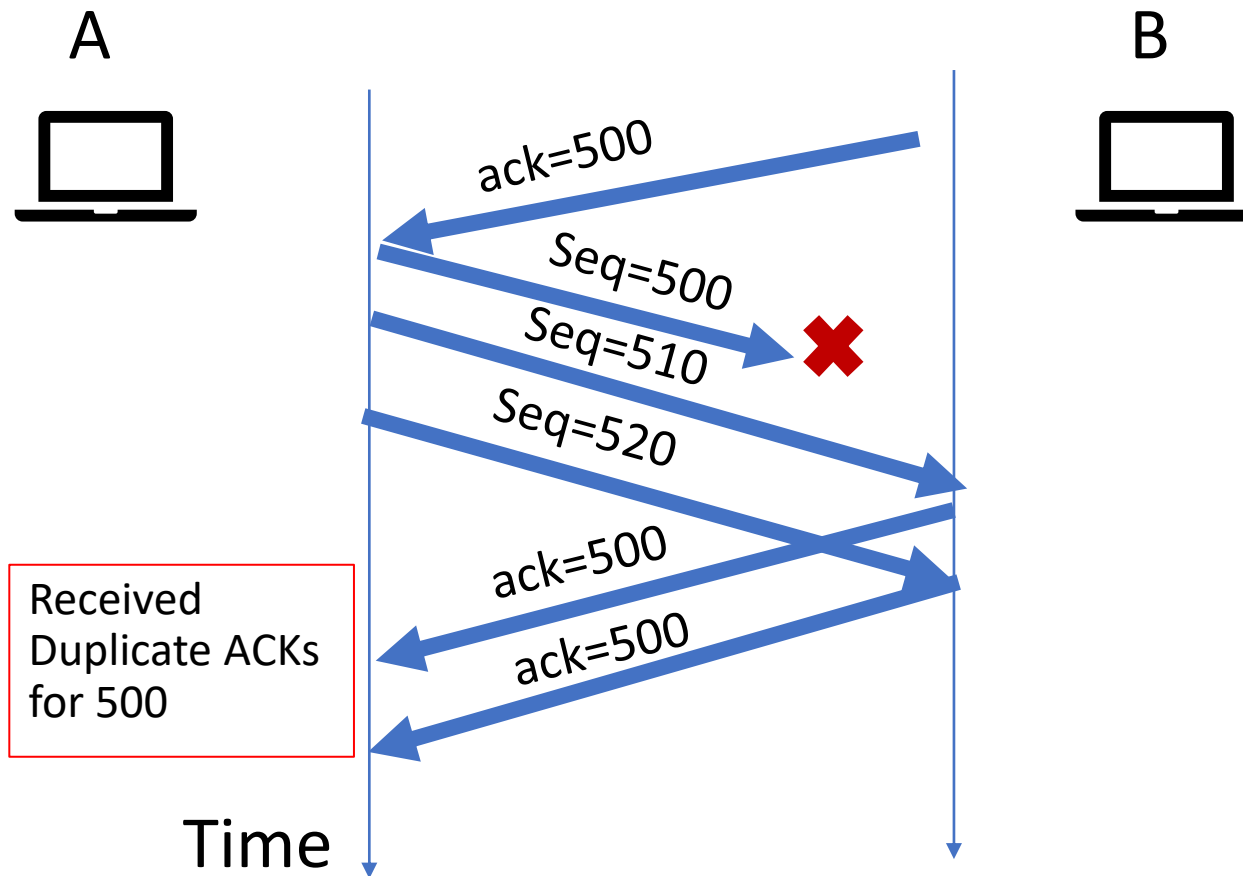
Aside – Message Exchange Diagram



When to Re-transmit - Timeout

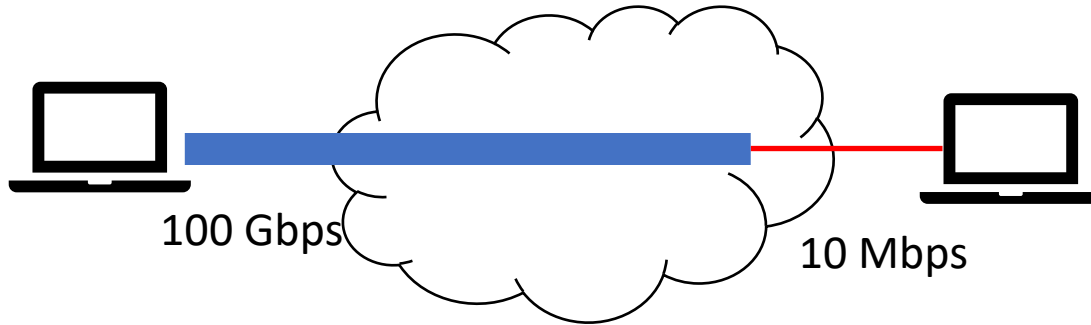


When to Re-transmit – Duplicate Ack



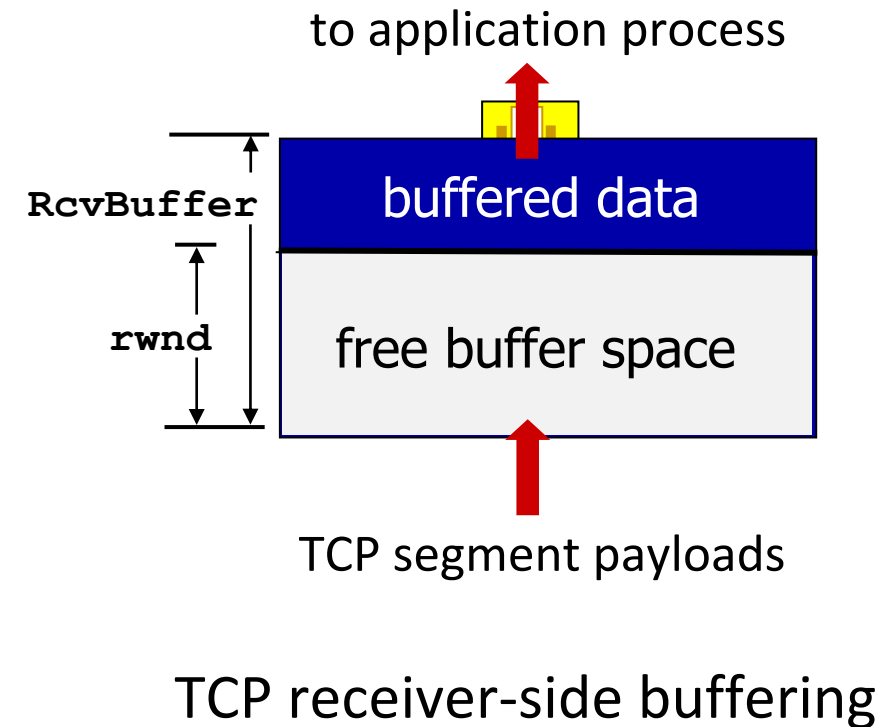
What about drops at the Receiver?

- Sender may be able to send at higher rates than receiver can accept
- Goal: Make it so the sender cannot overwhelm the receiver



Flow Control

- TCP receiver “advertises” free buffer space in rwnd field in TCP header
- sender limits amount of unACKed (“in-flight”) data to received rwnd



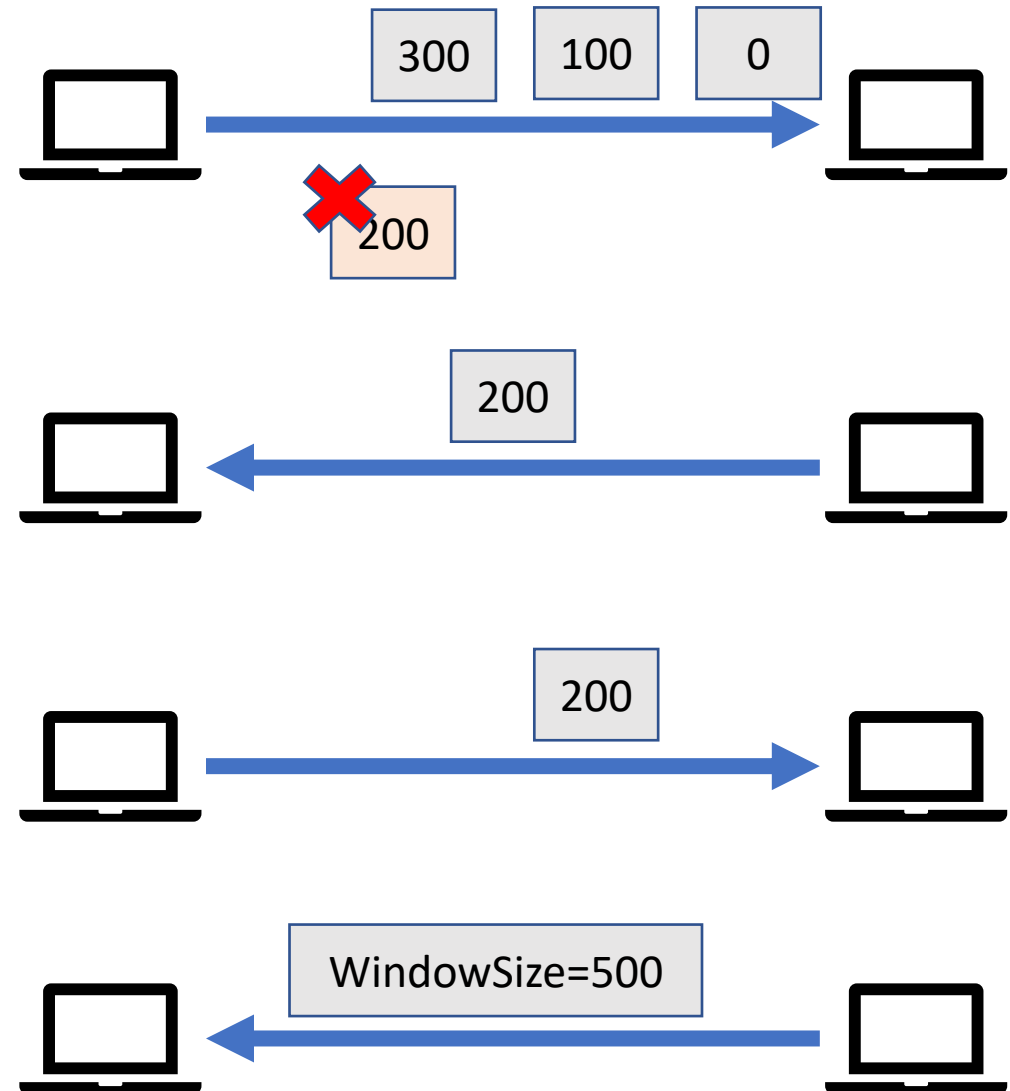
Flow Control – TCP Header

- Window Size field –how many bytes the host is currently able to receive

| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
|---------|-----|--|---|---|---|------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0000 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Summary – Reliable Transfer

- Sequence Number
- Acknowledgement Number
- Re-transmission
- Flow Control





University of Colorado **Boulder**

TCP Connection Establishment

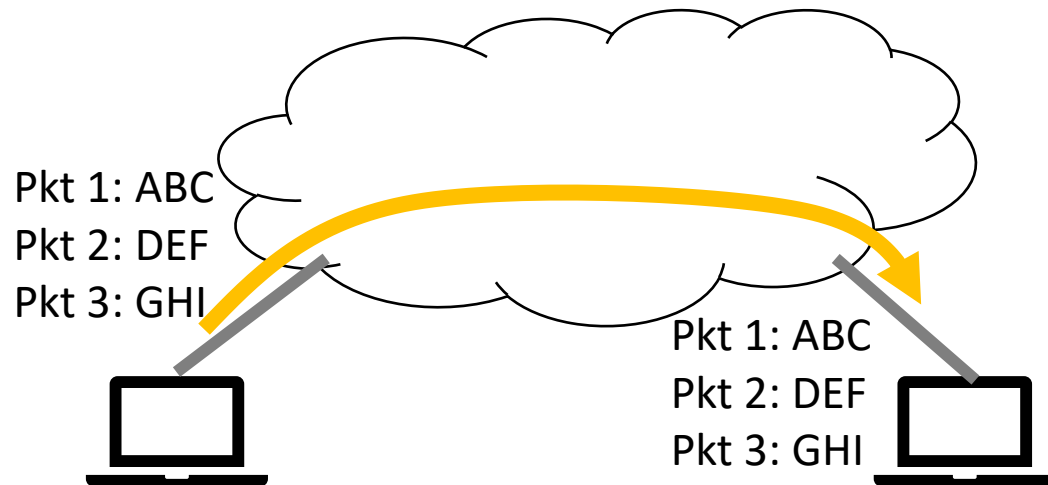
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

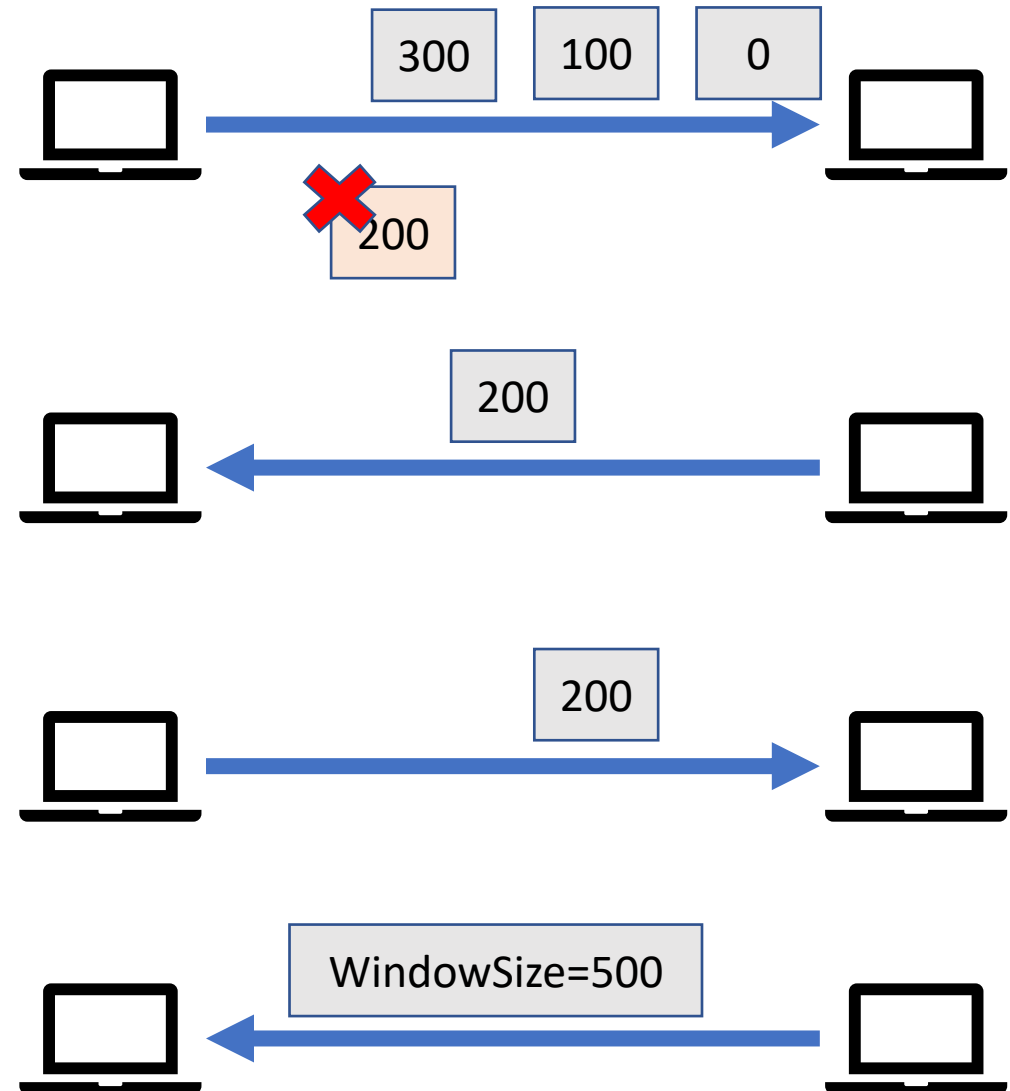
Problem 2: Reliable Transfer

- Goal - Allow an application to be assured that what they send is what is received (same data, same order).

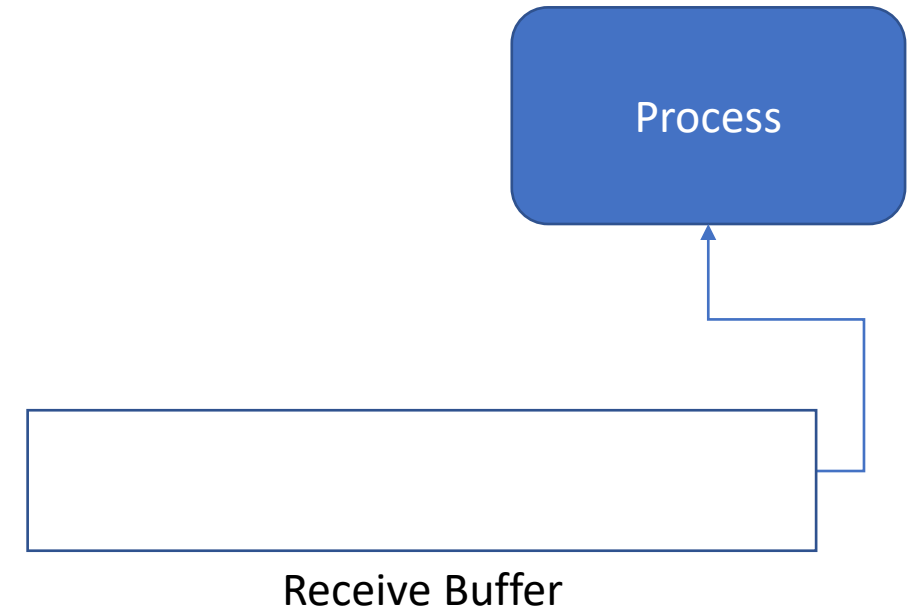
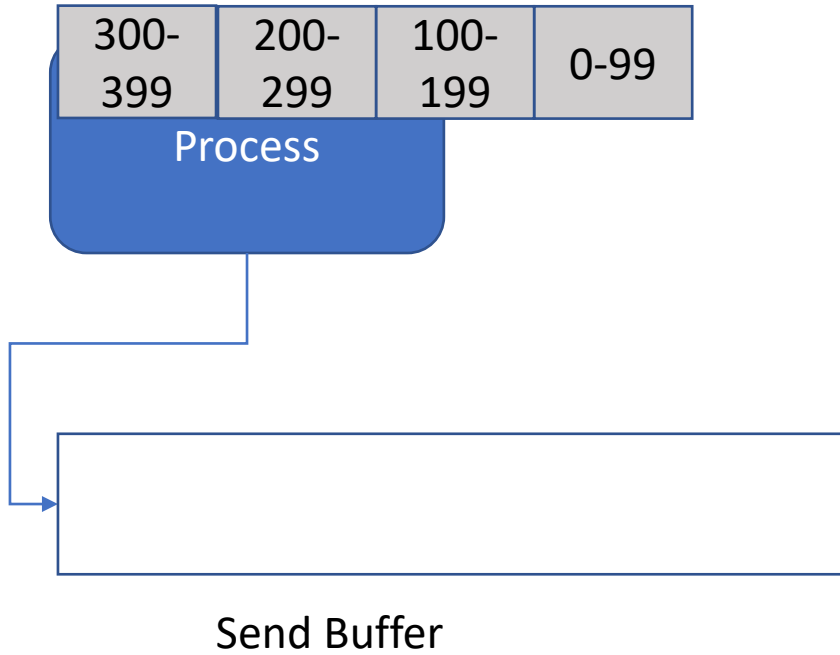


Summary – Reliable Transfer

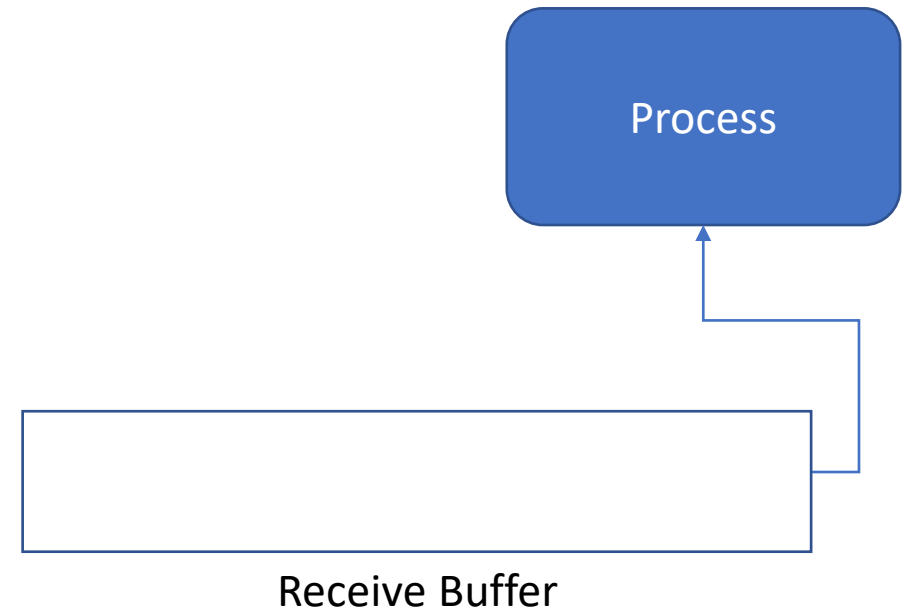
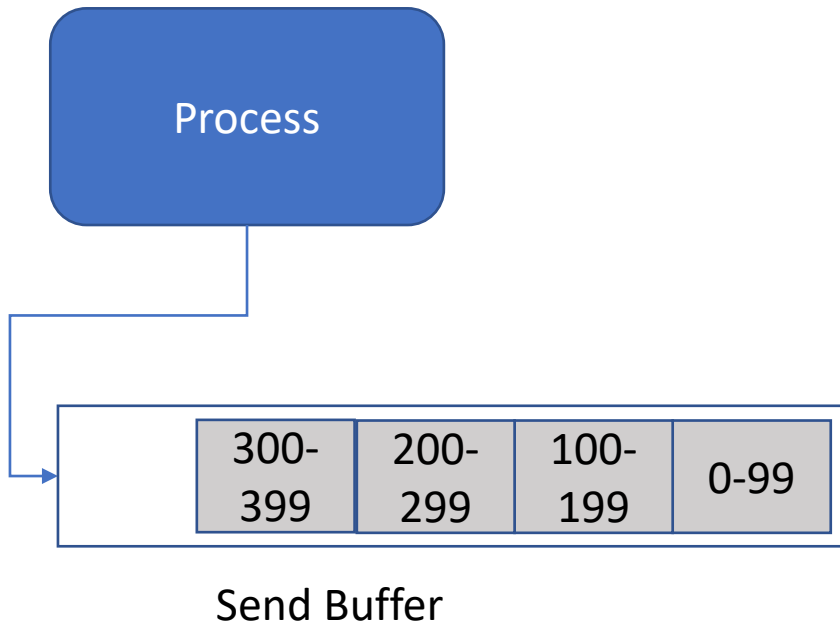
- Sequence Number
- Acknowledgement Number
- Re-transmission
- Flow Control



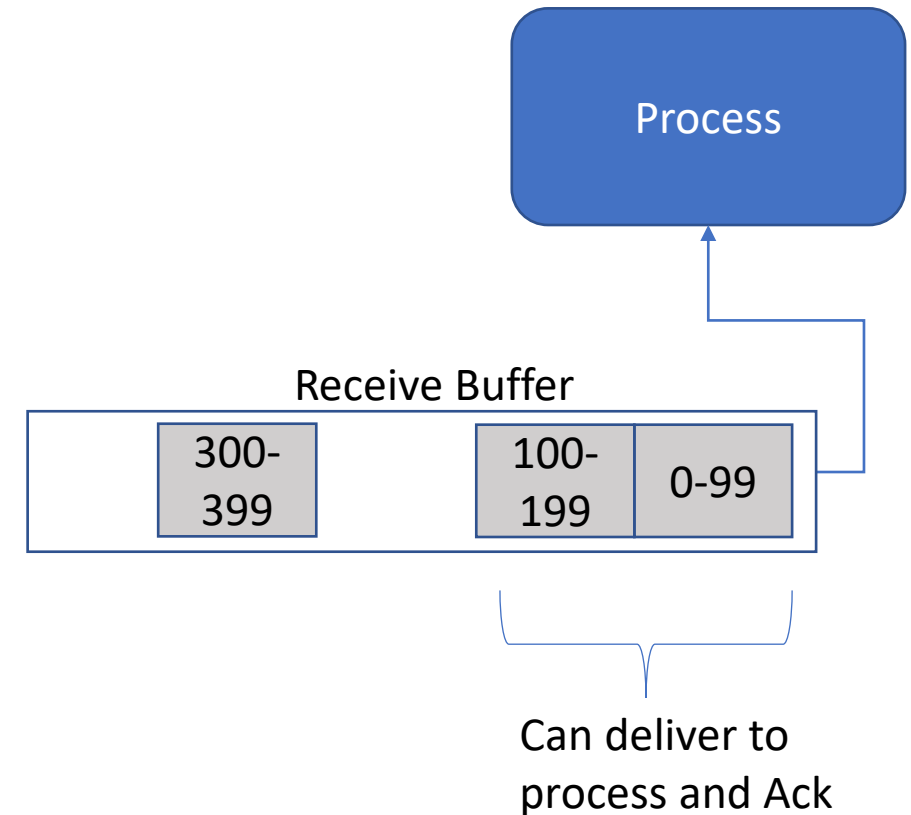
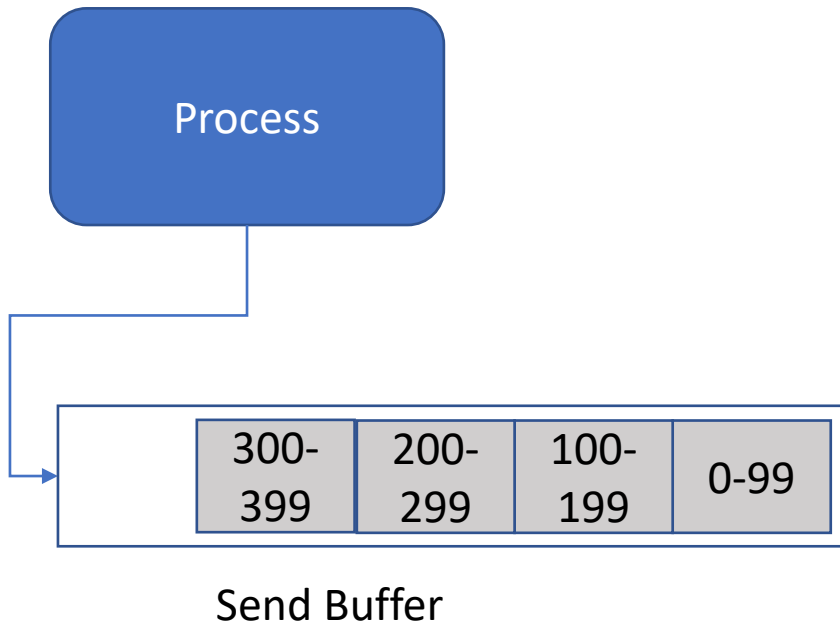
Each Side Needs to Store State



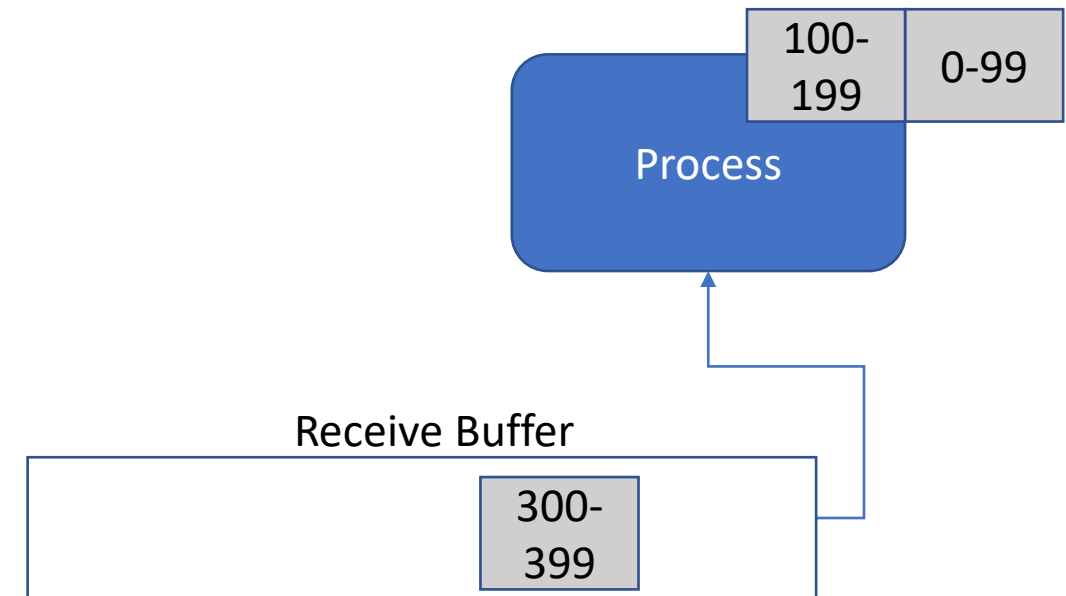
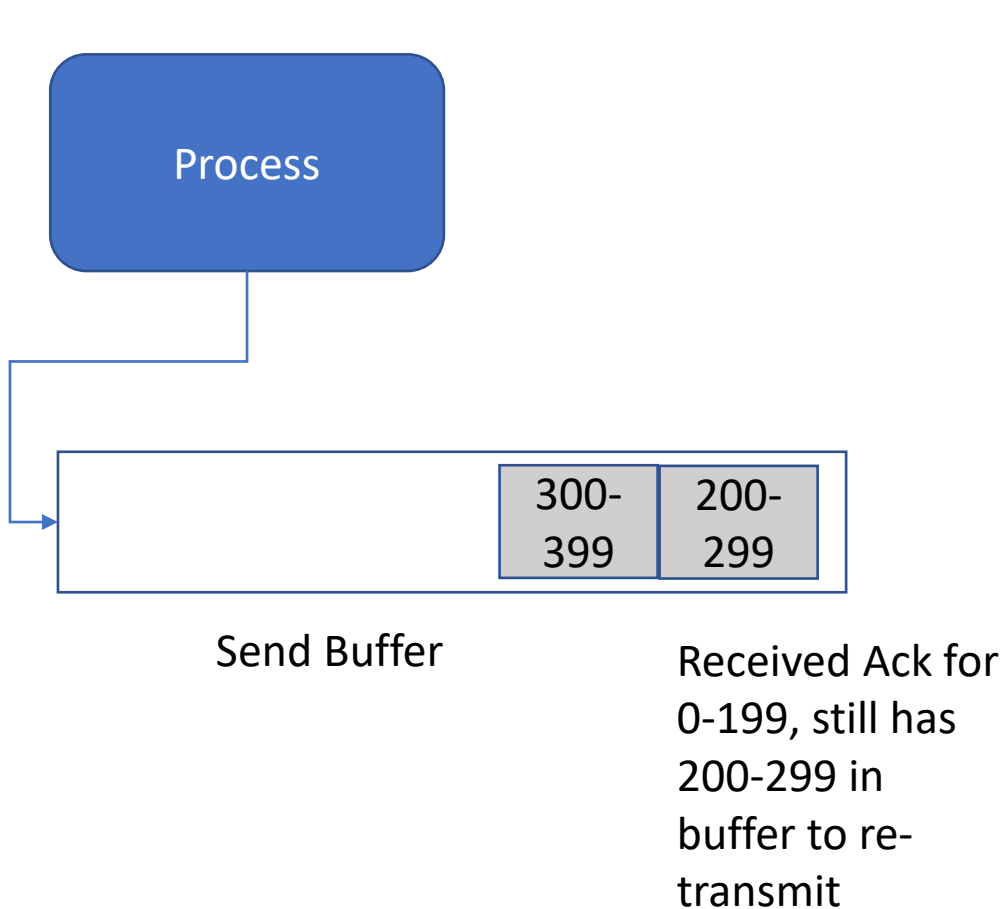
Each Side Needs to Store State



Each Side Needs to Store State



Each Side Needs to Store State



Connection Establishment

- TCP is connection oriented
- Connection establishment is an exchange of messages where each process agree to form a connection
- Uniquely identified by (src IP, dest IP, src Port, dest Port)
- Three way handshake
Basically: A to B) I'd like to connect, B to A) Ok, A to B) Thanks.

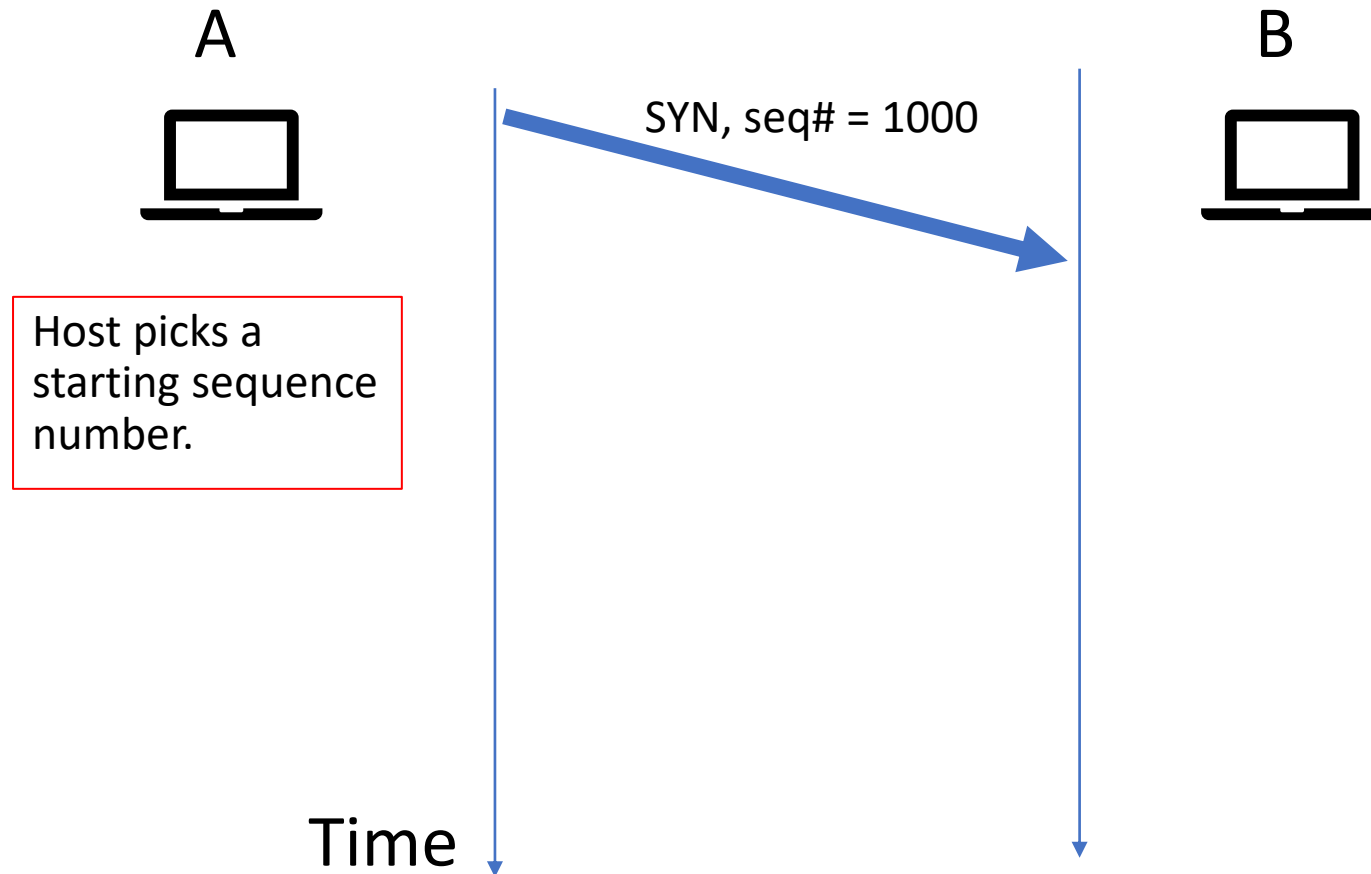


TCP Flags Used in Connection Establishment

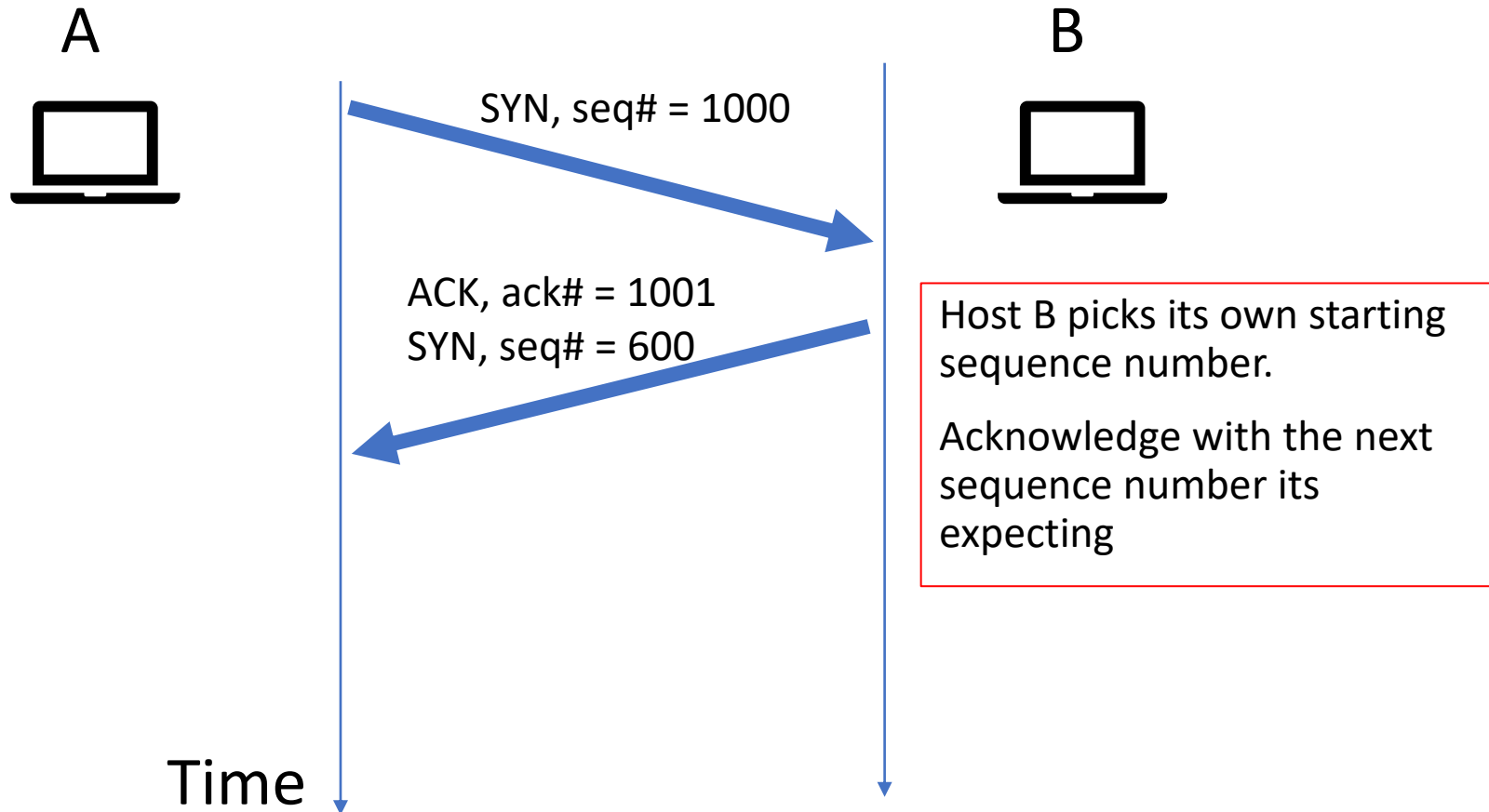
- SYN – synchronize sequence number
- ACK – the acknowledgement number field is valid

| Offsets | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
|---------|-----|--|---|---|---|------------------|---|---|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 64 | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 96 | Data offset | | | | Reserved 0000 | | | | C W R | E C E | U R G | A C K | P S H | R S T | S Y N | F I N | Window Size | | | | | | | | | | | | | | | |
| 16 | 128 | Checksum | | | | | | | | | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 20 | 160 | Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| : | : | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 56 | 448 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

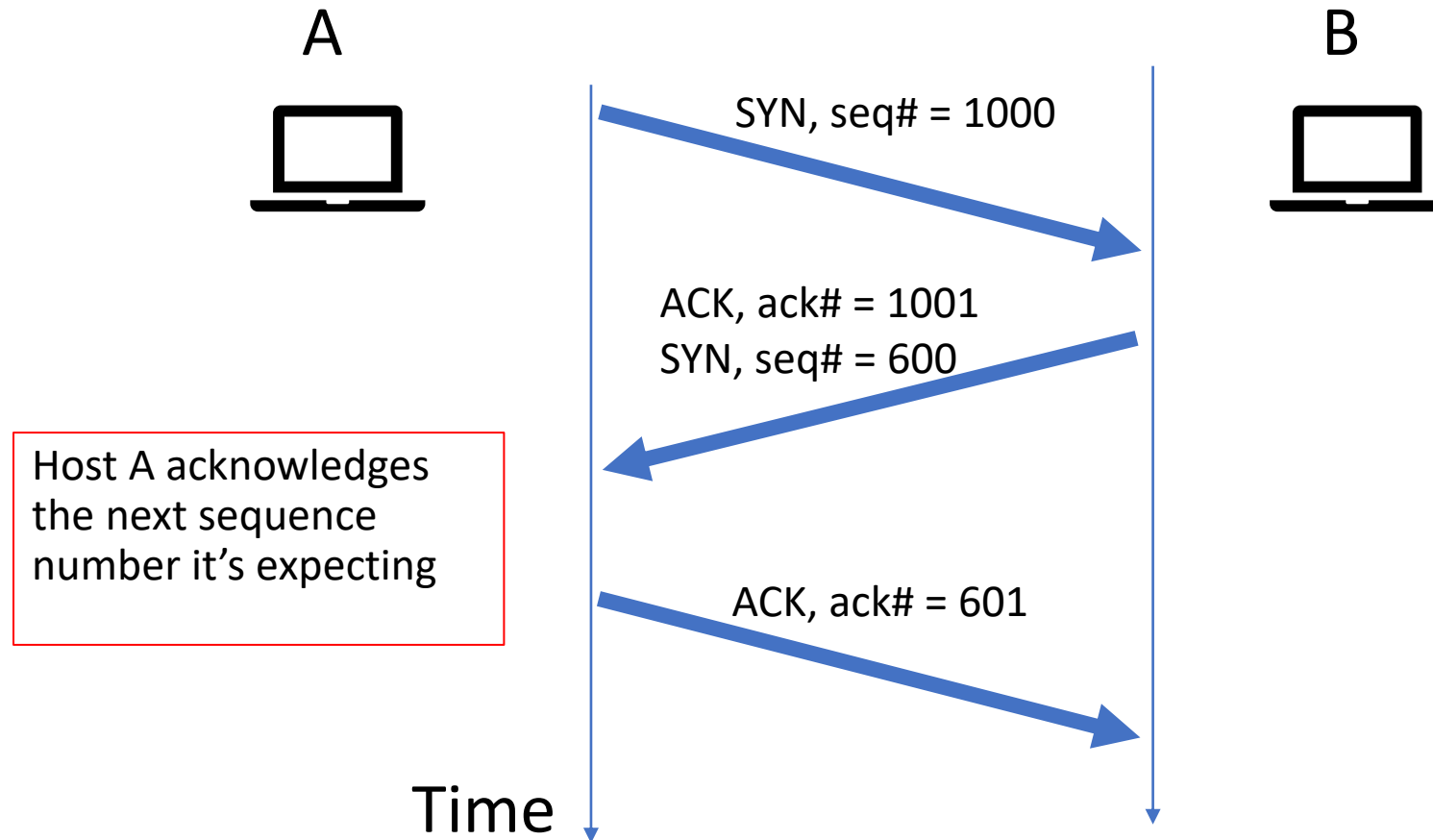
TCP Connection Establishment (3-way handshake)



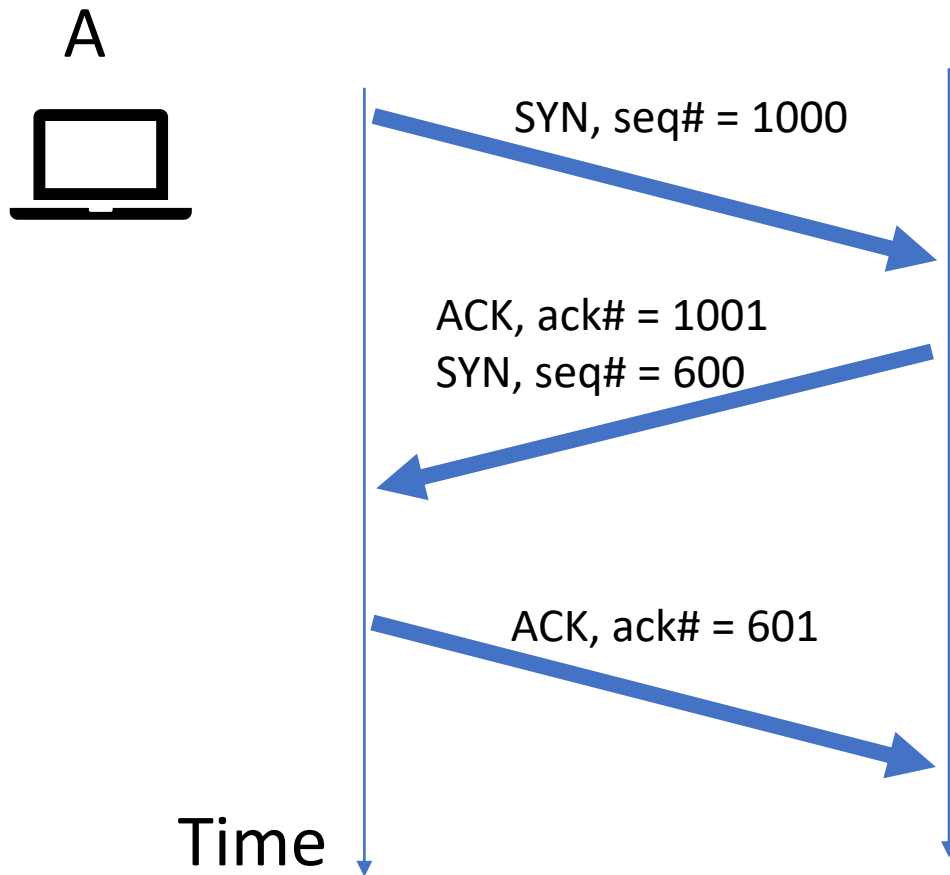
TCP Connection Establishment (3-way handshake)



TCP Connection Establishment (3-way handshake)



What if Packets are Lost during Connection Establishment?



Pkt 1 lost - A won't get an ACK, will resend

Pkt 2 lost - A won't get an ACK (for pkt1), will resend pkt 1.
B will ACK re-transmitted pkt1

Pkt 3 lost - B won't get an ACK (for pkt2), will resend pkt 2.
A will ACK retransmitted pkt2





University of Colorado **Boulder**

Congestion Control

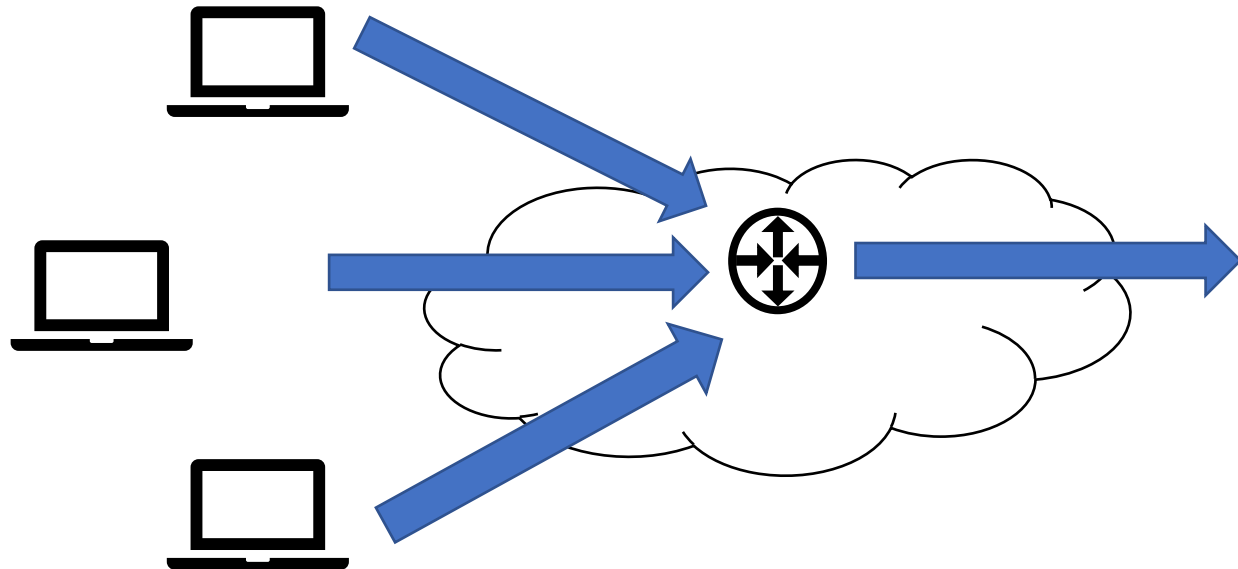
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

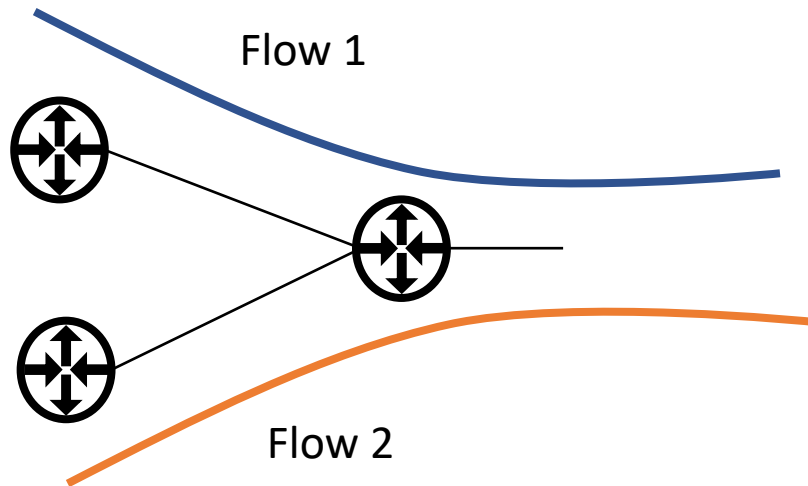
Problem 3: Congestion Control

- Congestion in the network will occur – leading to packets being dropped
- Goal: What if senders could detect this and backoff as needed?

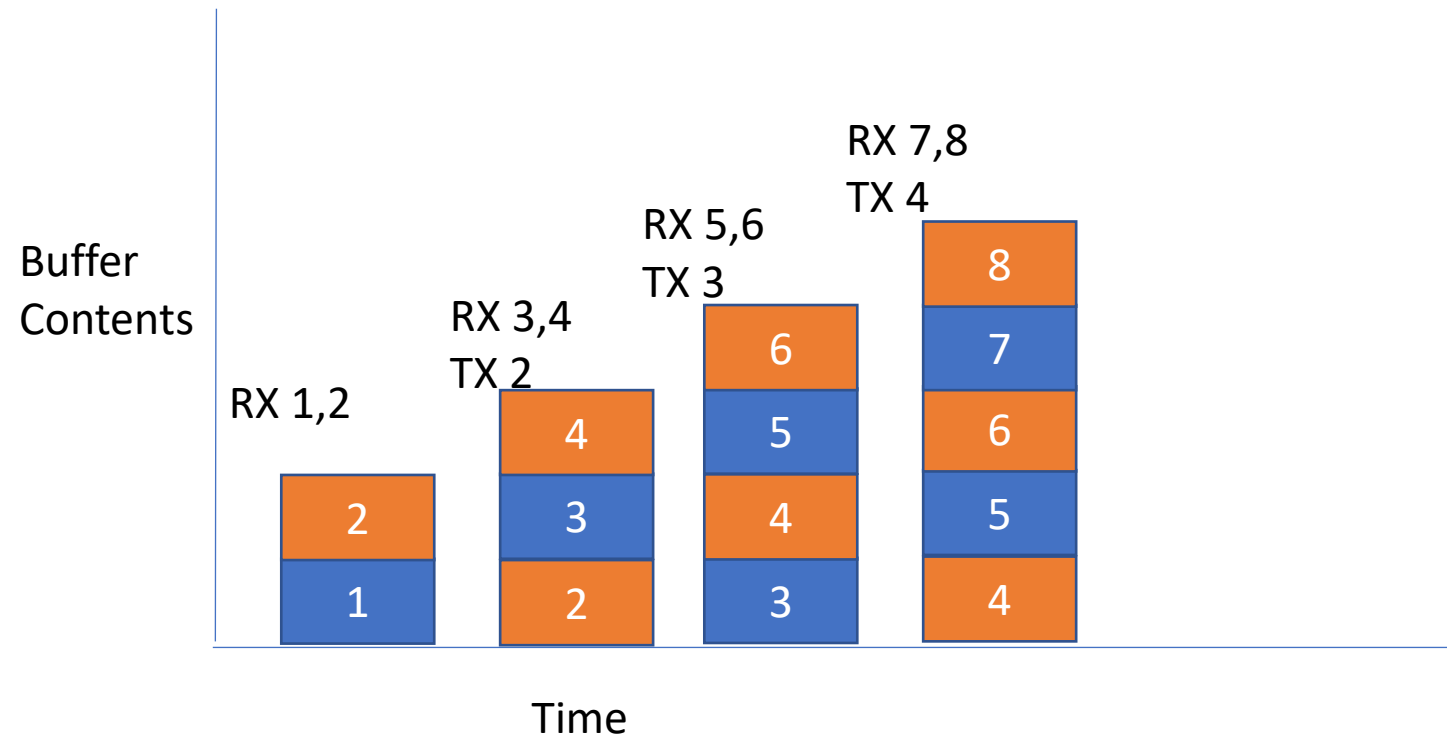


How Congestion Happens

- Send rate out of a link lower than the receive rate
(e.g., different bandwidths, or forwarding from multiple ports to one)

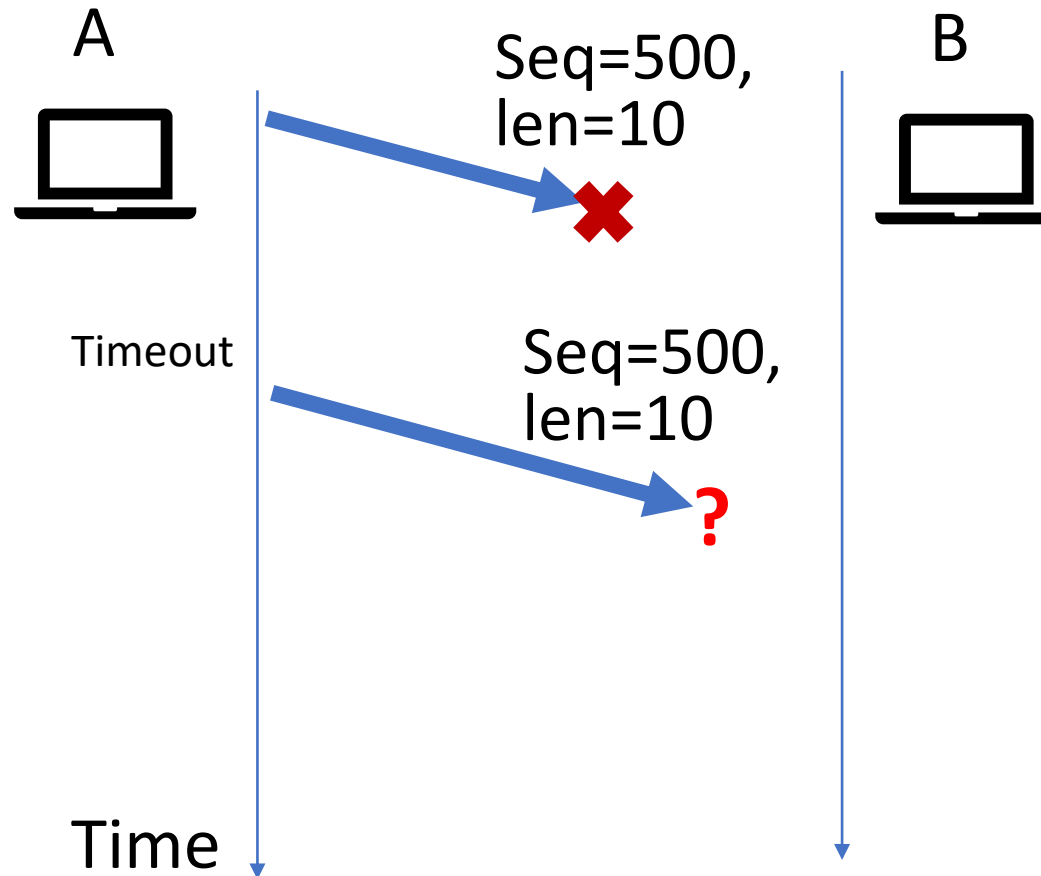


Each link can transfer 1 pkt per unit time



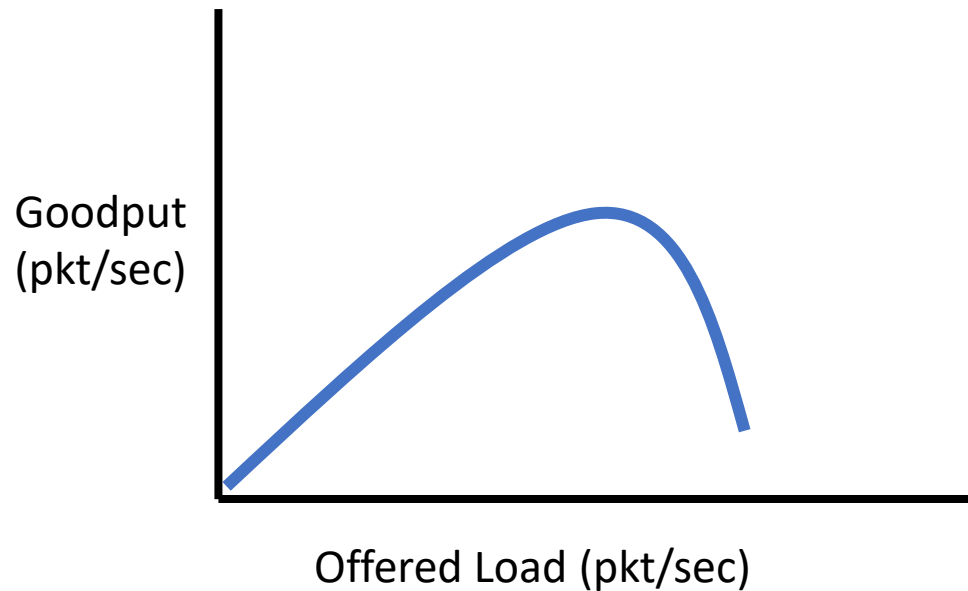
What Happens When Congestion Occurs

- Packets get dropped (by router)
- Then re-transmitted (by host)



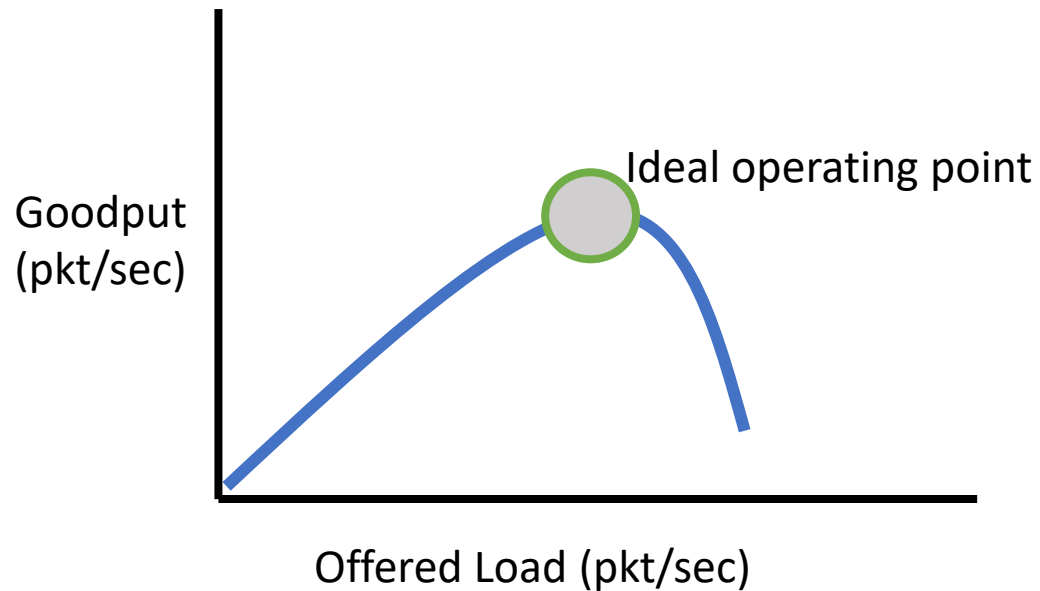
End Result – Congestion Collapse

- When nearing capacity, goodput decreases rapidly
- Buffers get full, but sending keeps overloading them, so more and more packets getting dropped and re-transmitted



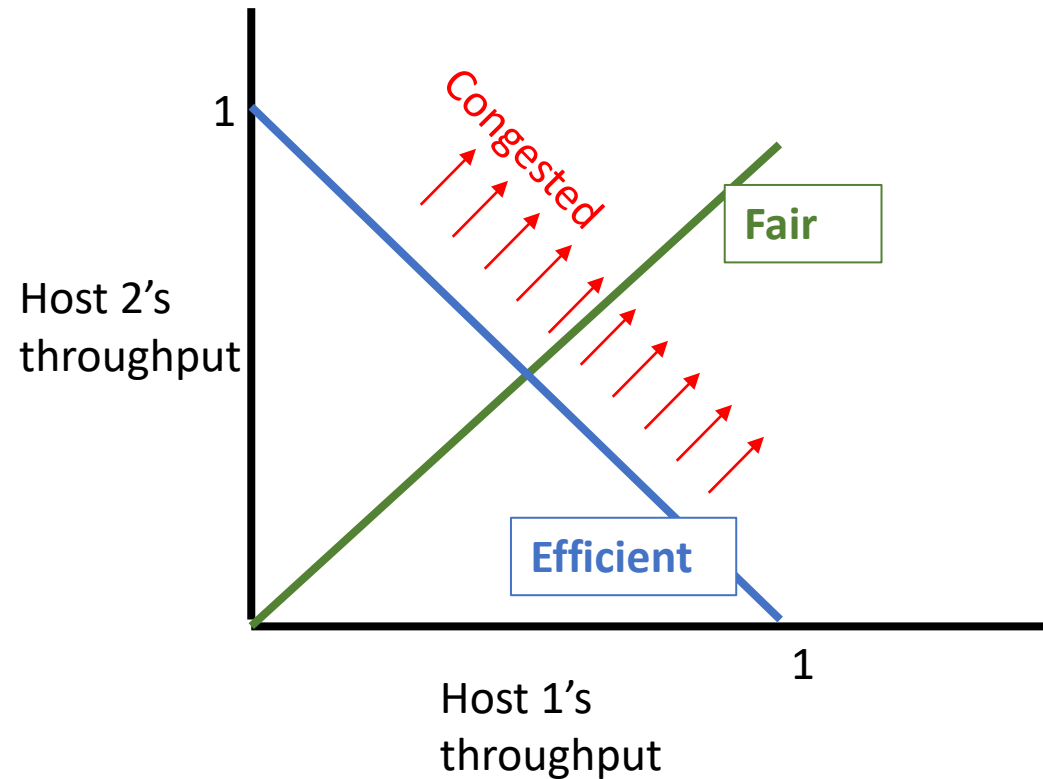
Congestion Control

- Can we get the network to operate just before it collapses?
- Goal – efficient, fair



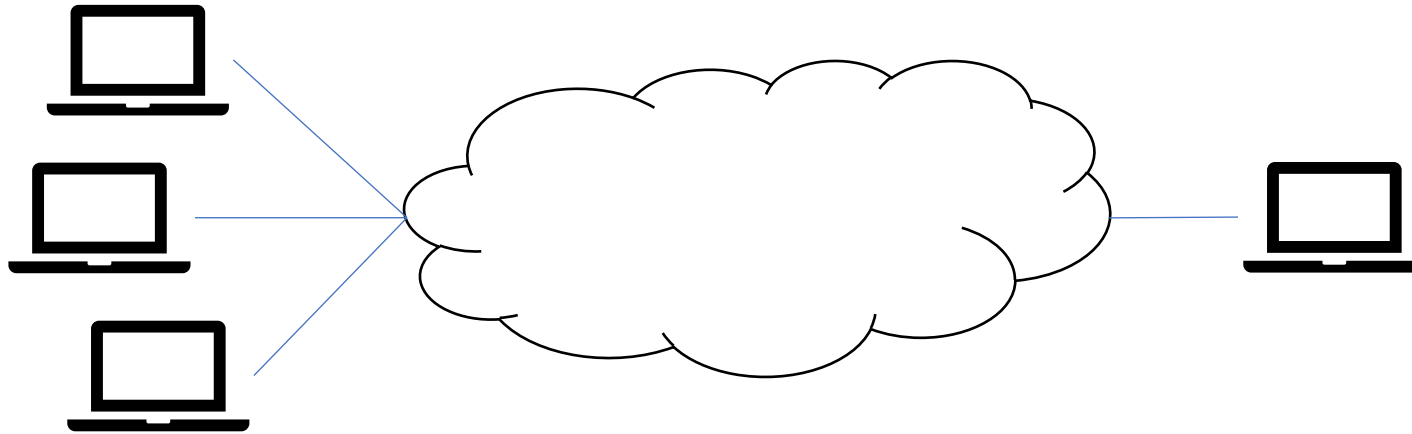
Fair and Efficient Illustrated

- Fair – each flow receives an equal share of the bandwidth of a link
- Efficient – full bandwidth of link is used



Resource Allocation Approaches

- Reservation – in network, reserve bandwidth on each router
- Feedback and Adjust – on hosts, detect congestion and adjust send rate



TCP Congestion Control Overview

Senders can increase sending rate until congestion occurs, then decrease sending rate

- Feedback – implicit
- Adjust – window-based





University of Colorado **Boulder**

TCP Congestion Control

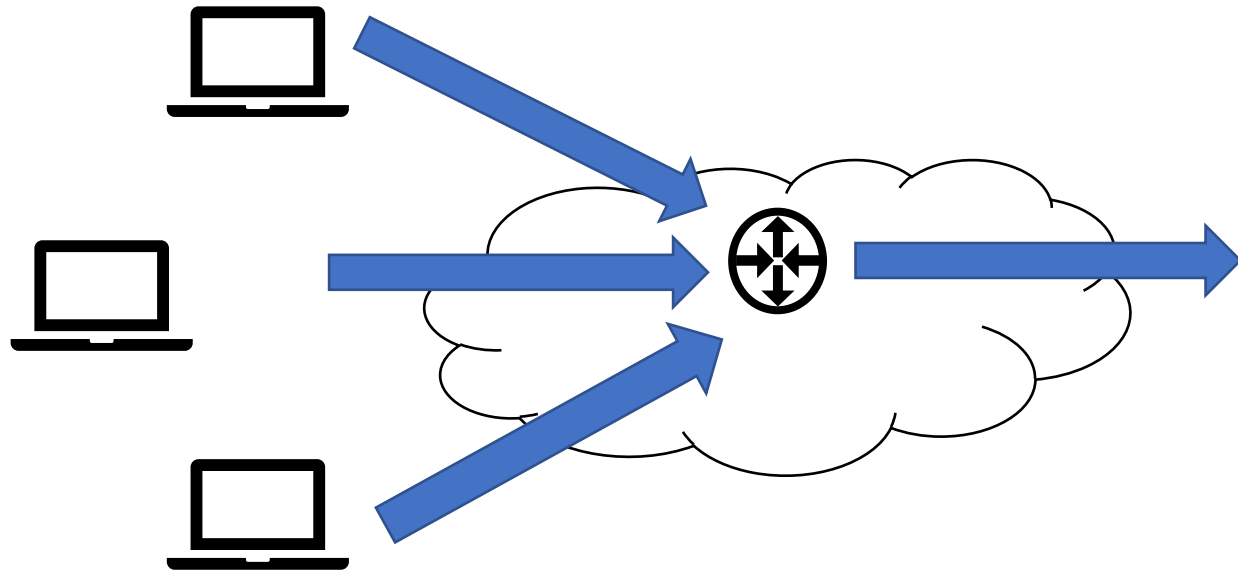
Course: Networking Fundamentals
Module: Transport



University of Colorado **Boulder**

Problem 3: Congestion Control

- Congestion in the network will occur – leading to packets being dropped
- Goal: What if senders could detect this and backoff as needed?



TCP Congestion Control Overview

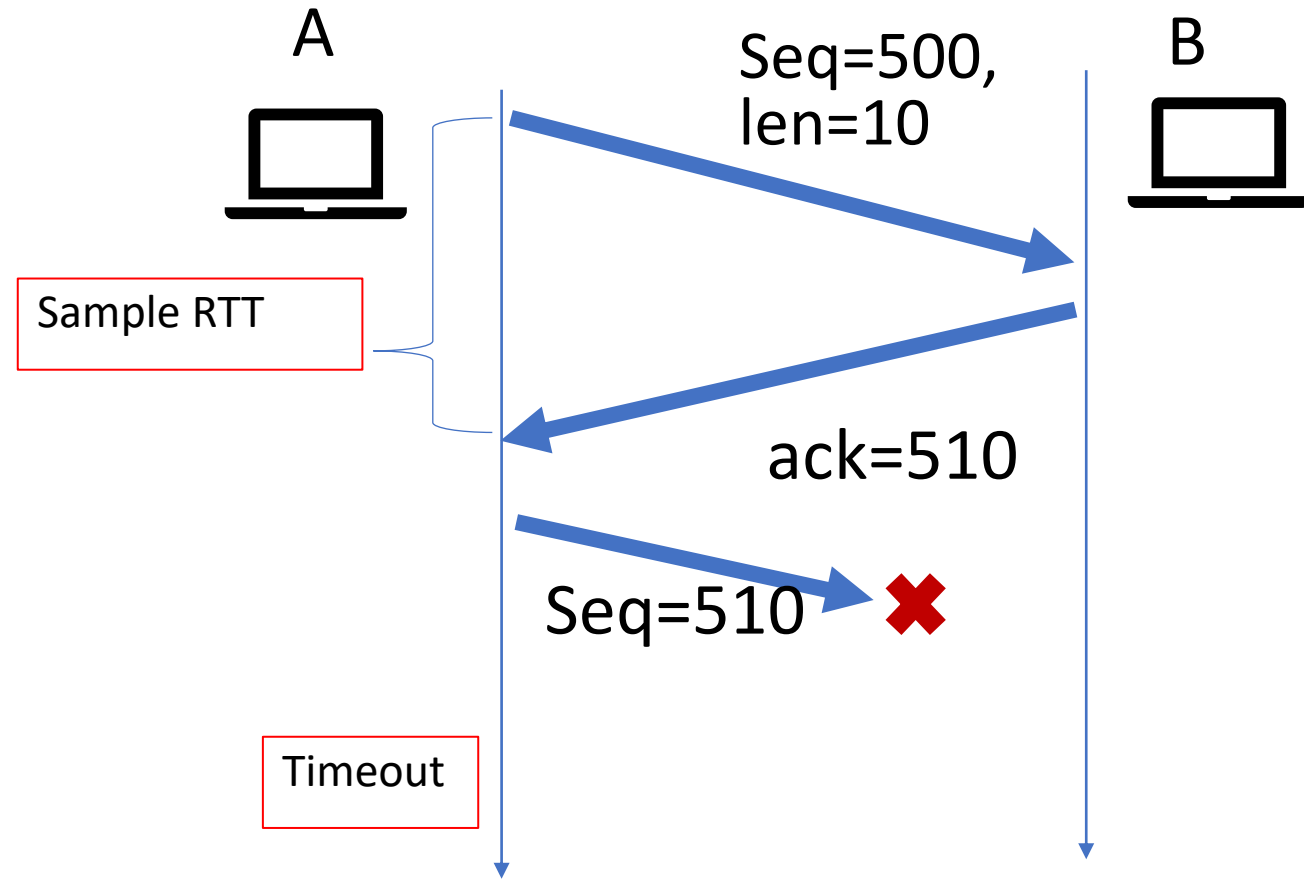
Senders can increase sending rate until congestion occurs, then decrease sending rate

- Feedback – implicit
- Adjust – window-based



Feedback

- Packet Loss
 - Timeout
 - Duplicate ACK
- Packet Delay
- Explicit Congestion Notification (requires router support)



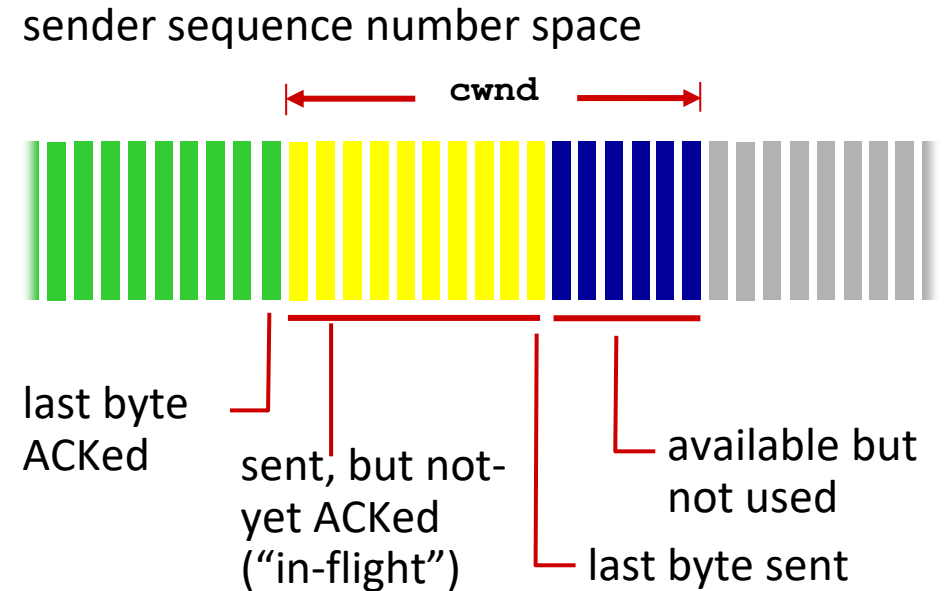
Window

- Host limits sending through a Congestion Window (cwnd)

$$\text{lastByteSent} - \text{lastByteAck} \leq \text{cwnd}$$

Note: recall flow control – window size TCP header field. (rwnd)

cwnd is internal state on host



Adjusting Congestion Window

- Additive Increase Multiplicative Decrease (AIMD)
- Shown to:
 - optimize congested flow rates network wide!
 - have desirable stability properties

Additive Increase

increase sending rate by 1 maximum segment size every RTT until loss detected

Multiplicative Decrease

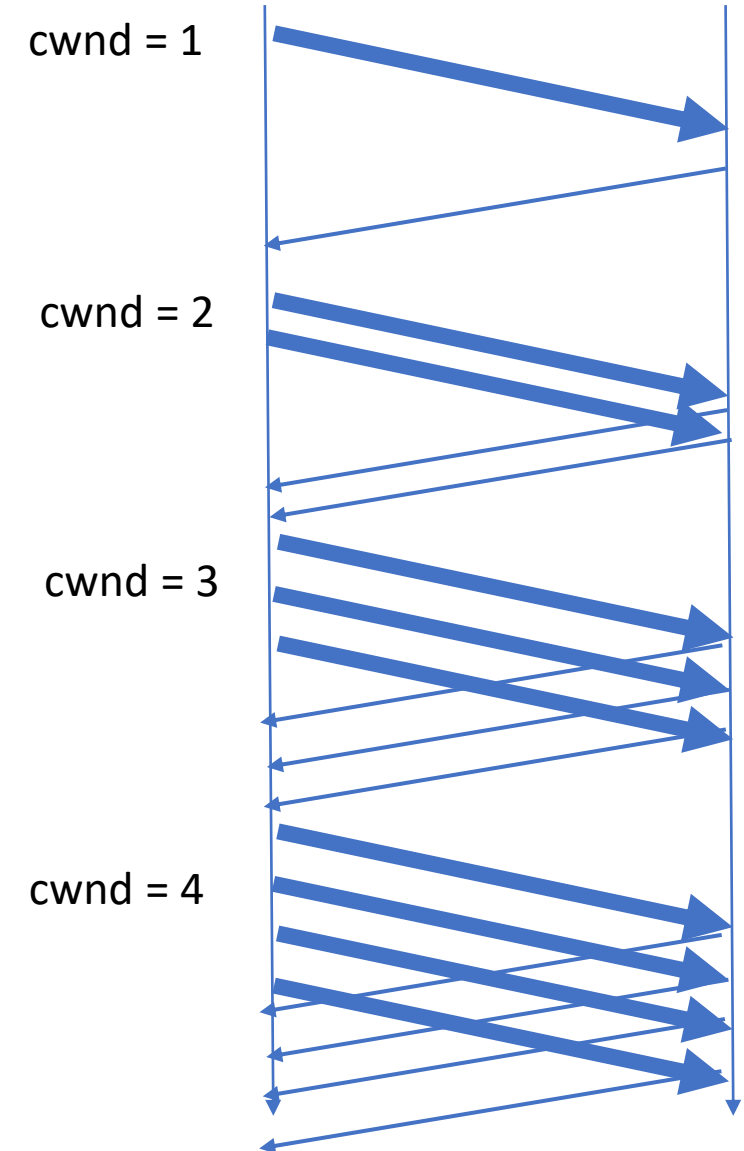
decrease sending rate by some multiplicative factor (e.g., by half) when loss detected



Additive Increase

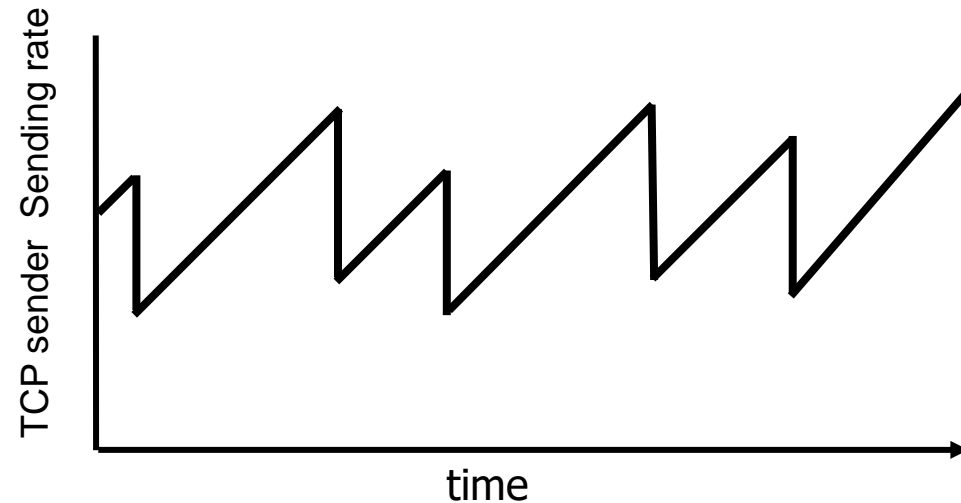
- Timeline with increase cwnd by MSS for each RTT (but, do it partially with each ACK)

$\text{Increment} = \text{MSS} \times (\text{MSS} / \text{CongestionWindow})$
 $\text{CongestionWindow} += \text{Increment}$



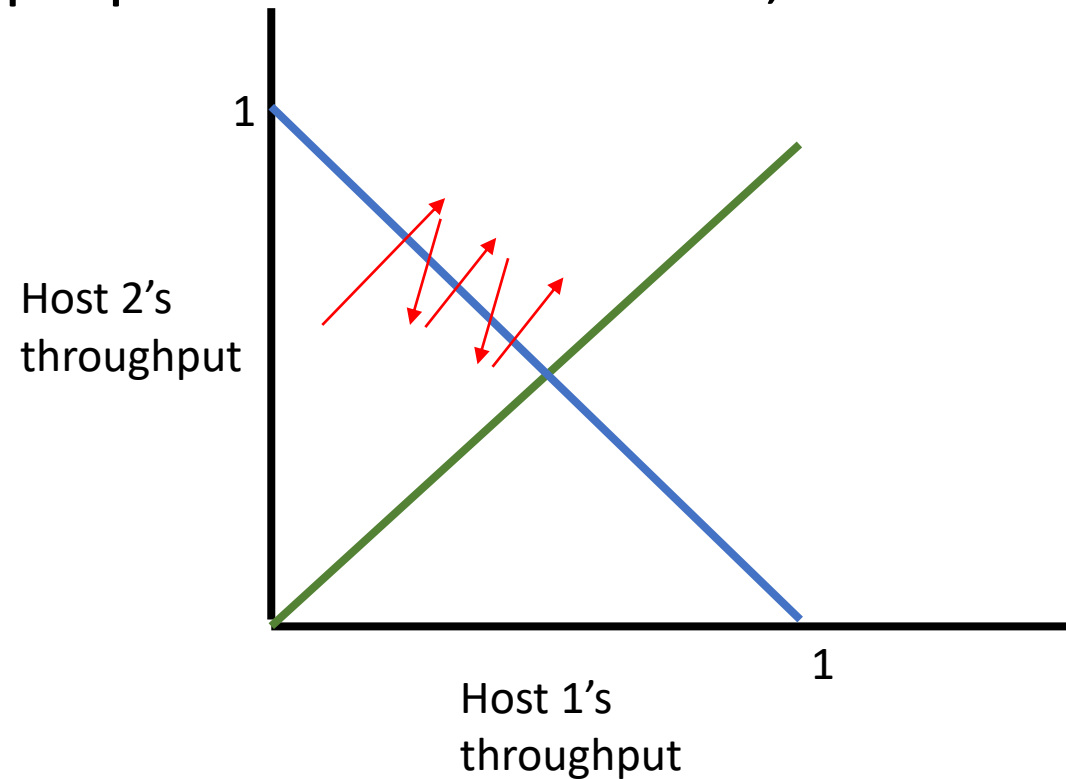
Multiplicative Decrease

- When detect congestion, decrease send rate
- Leads to sawtooth pattern
 - AI (probe for bandwidth), MD (rapidly correct)
- Multiplicative has been shown to be a necessary condition for stability



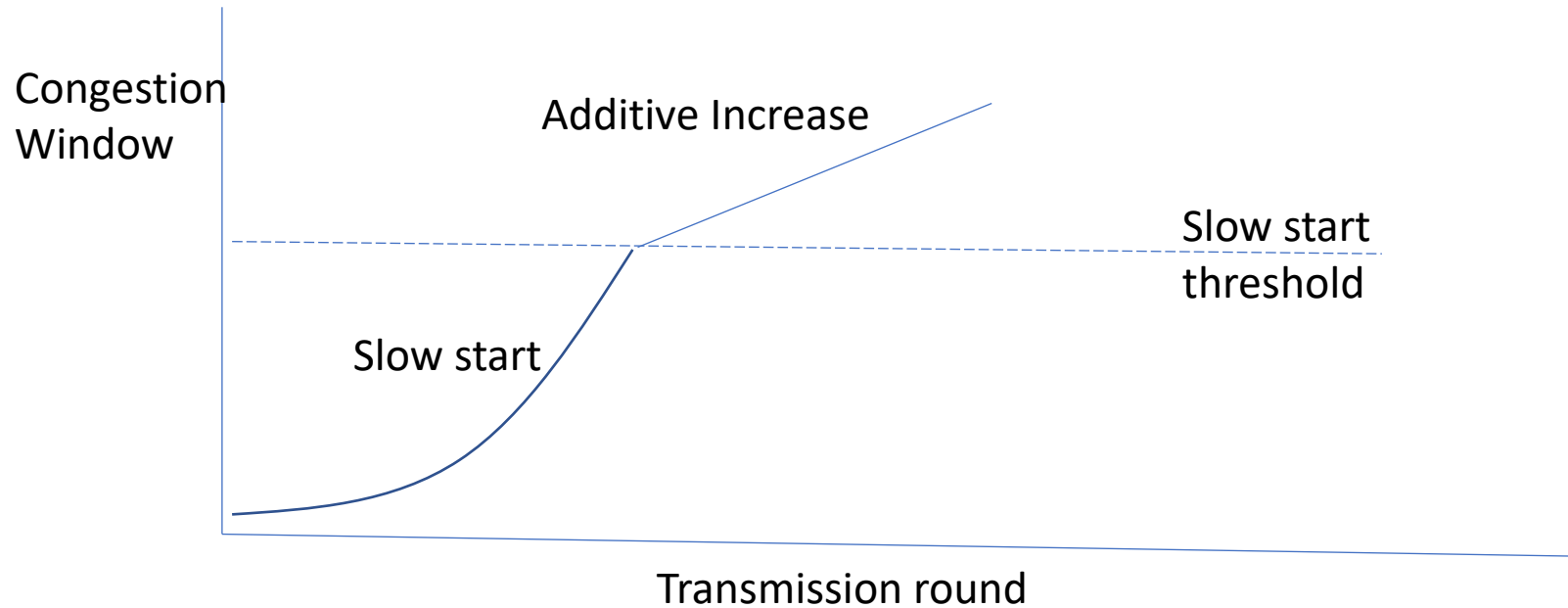
Why AIMD Conceptually Works

- Increase is linear (all increase the same)
- Decrease is proportional to send rate, so converge towards fair line



TCP Slow Start

- Additive increase is too conservative (too slow to get to congestion)
- TCP Slow Start - Increase cwnd every Ack (instead of every RTT)





University of Colorado **Boulder**