

Congestion Control



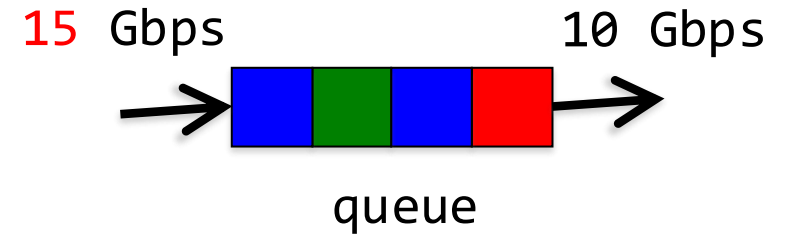
COS 316: Principles of Computer System Design

Lecture 7

Wyatt Lloyd

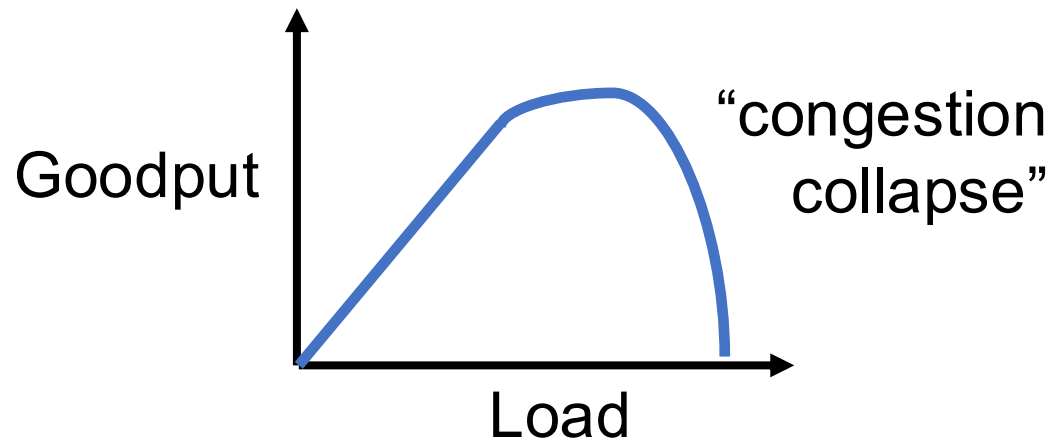
Congestion

- Best-effort network does not “block” calls
 - So, they can easily become overloaded
 - Congestion == “Load higher than capacity”
- Examples of congestion
 - Link layer: Ethernet frame collisions
 - Network layer: full IP packet buffers
- Excess packets are simply dropped
 - And the sender can simply retransmit



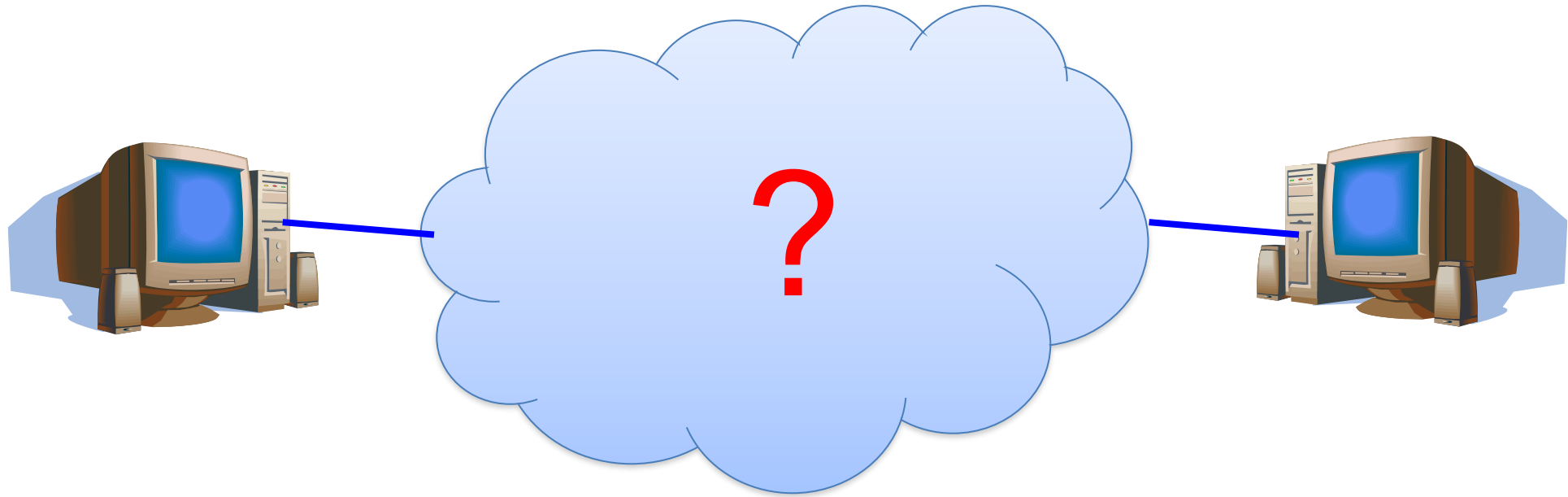
Congestion Collapse

- Easily leads to *congestion collapse*
 - Senders retransmit the lost packets
 - Leading to even *greater* load
 - ... and even *more* packet loss



Increase in load
that results in a
decrease in useful
work done.

Detect and Respond to Congestion



- What does the end host see?
- What can the end host change?

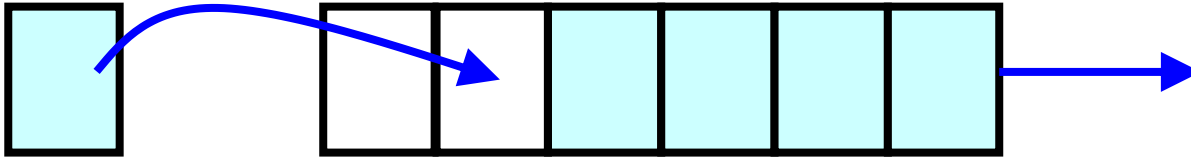
Detecting Congestion

- Transport layer
 - Observing end-to-end performance
 - Packet delay or loss over the path

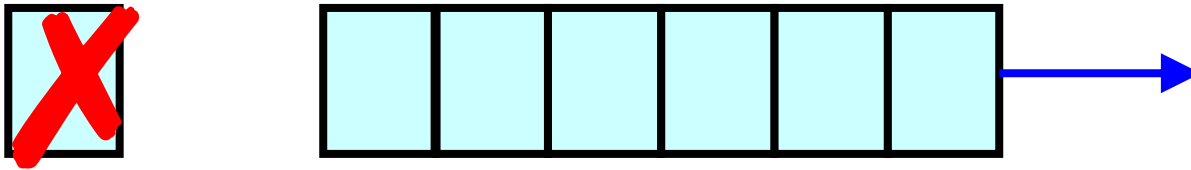
TCP Congestion Control

Congestion in a Drop-Tail FIFO Queue

- Access to the bandwidth: first-in first-out queue
 - Packets transmitted in the order they arrive

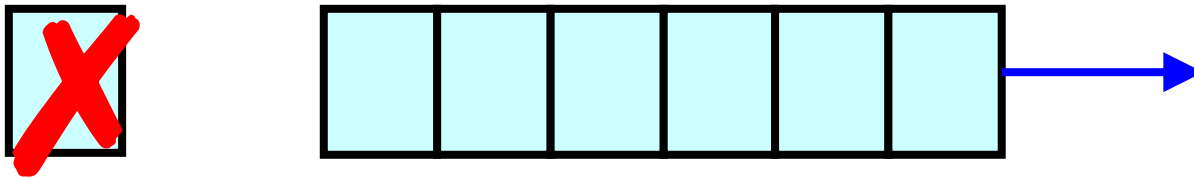


- Access to the buffer space: drop-tail queuing
 - If the queue is full, drop the incoming packet



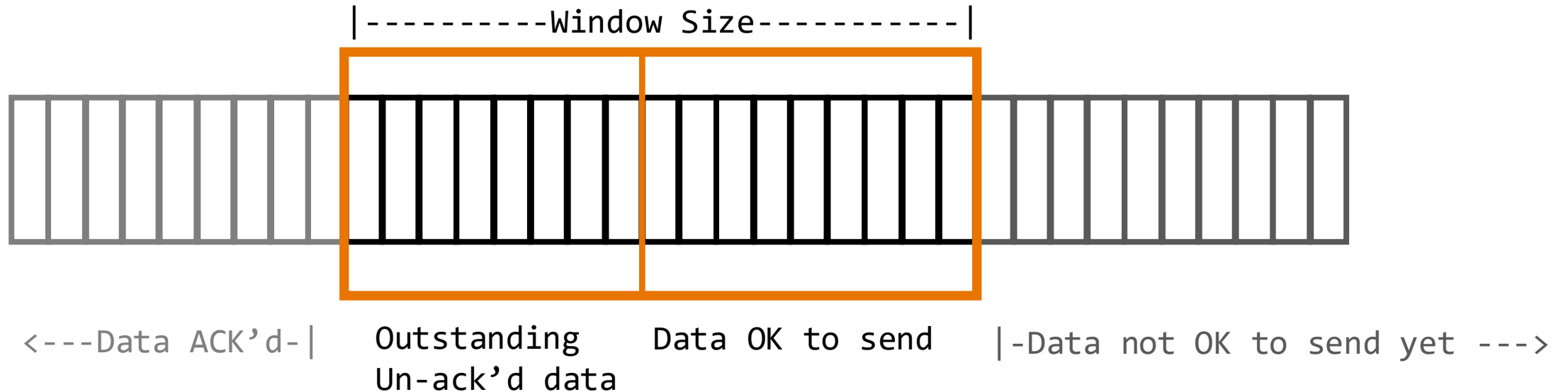
How it Looks to the End Host

- Delay: Packet experiences high delay
- Loss: Packet gets dropped along path
- How does TCP sender learn this?
 - **Delay:** Round-trip time estimate
 - **Loss:** Timeout and/or duplicate acknowledgments



TCP Congestion Window

- Each TCP sender maintains a congestion window
 - Max number of bytes to have in transit (not yet ACK'd)



- Limits the sending rate of traffic

Receiver Window vs. Congestion Window

- Flow control
 - Keep a **fast sender** from overwhelming a **slow receiver**
- Congestion control
 - Keep a **set of senders** from overloading **the network**
- Different concepts, but similar mechanisms
 - TCP flow control: receiver window
 - TCP congestion control: congestion window
 - Sender TCP window = **min { congestion window, receiver window }**

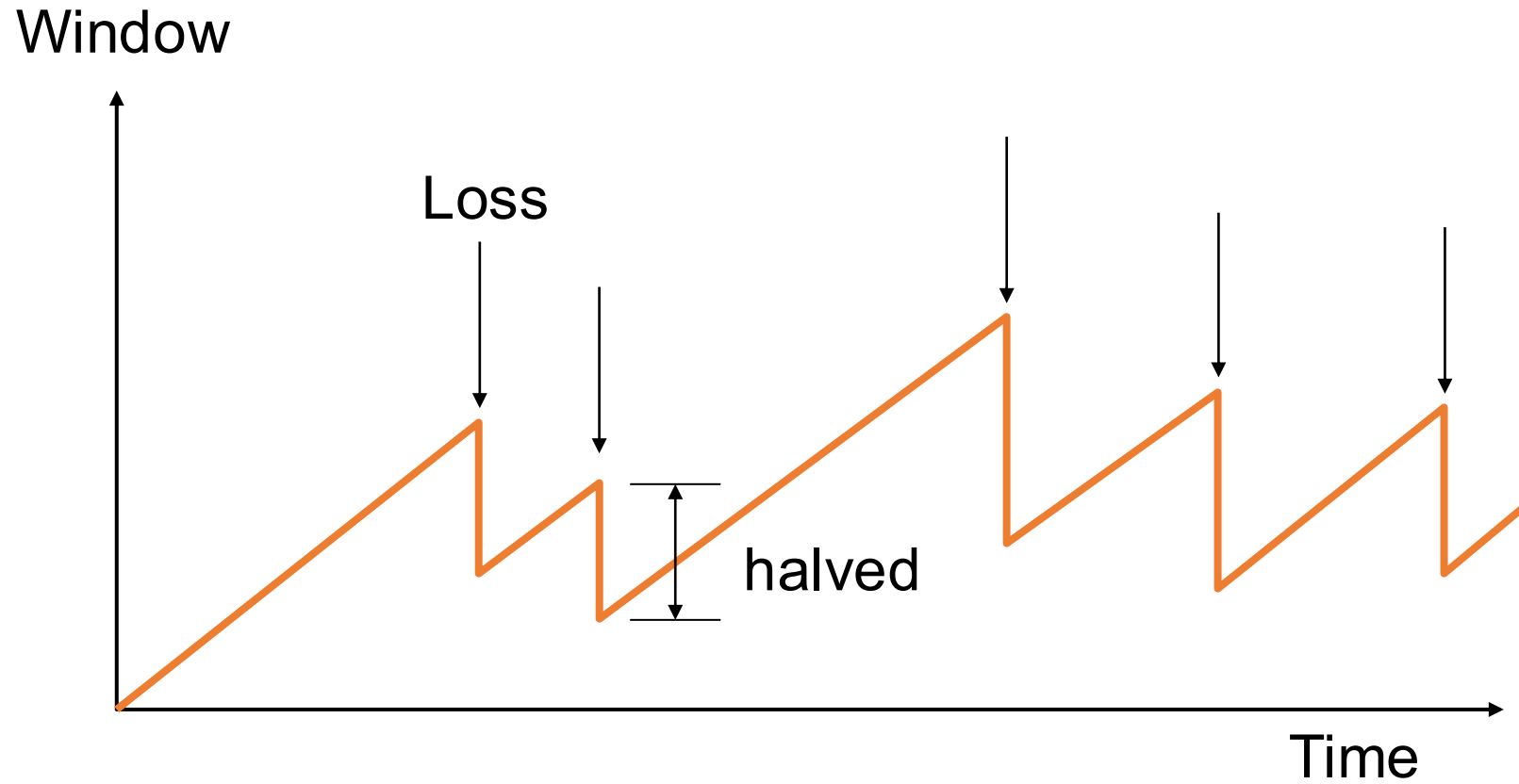
TCP Sender Adjusts the Congestion Window

- Packet loss (fail!)
 - Suspect an overutilized network (congestion)
 - Pessimistically decrease the congestion window
- Packet delivery (succeed!)
 - Suspect an underutilized network
 - Optimistically increase the sending rate
- Always struggling to find the right rate
 - Pro: avoids the need for explicit feedback
 - Con: continually under-shooting and over-shooting

How Much Should the Sender Adapt?

- Additive increase (AI)
 - Cautious to avoid triggering congestion
 - On success of last window of data, increase congestion window by 1 packet
- Multiplicative decrease (MD)
 - Aggressive to respond quickly to congestion
 - On the loss of packet, divide congestion window in half
- Much quicker to slow down than speed up?
 - Over-sized windows (causing loss) are much worse than under-sized windows (causing lower throughput)

Leads to the TCP “Sawtooth”



Sources of Poor TCP Performance

- The below conditions *may* primarily result in:
(A) Higher packet latency (B) Greater loss (C) Lower throughput

1. Larger buffers in routers
2. Smaller buffers in routers
3. Smaller buffers on end-hosts
4. Slow application receivers

TCP seeks “Fairness”

Fair and Efficient Use of a Resource

- Suppose 3 users share the bandwidth on a single link
 - E.g., link has total of 30 Gbps
- What is a **fair** allocation of bandwidth?
 - Suppose user demand is unlimited
 - Allocate each a **1/n** share (e.g., 10 Gbps each)
- But, “equality” is not enough
 - Which allocation is best: [5, 5, 5] or [18, 6, 6]?
 - [5, 5, 5] is more “fair”, but [18, 6, 6] more efficient
 - What about [5, 5, 5] vs. [22, 4, 4]?

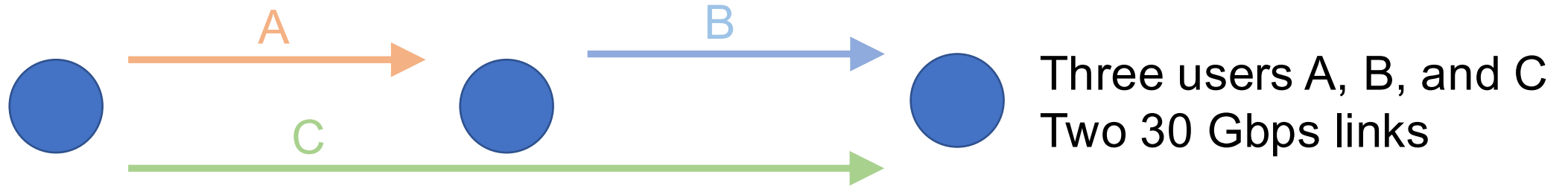
Fair Use of a Single Resource

- What if some users have limited demand?
 - E.g., 3 users where 1 user only wants 6 Gbps
 - And the total link capacity is 30 Gbps
- Should we still do an “equal” allocation?
 - E.g., [6, 6, 6]
 - But that leaves 12 Gbps unused
- Should we allocate in proportion to demand?
 - E.g., 1 user wants 6 Gbps, and 2 each want 20 Gbps
 - Allocate [4, 13, 13]?
- Or, give the least demanding user all they want?
 - E.g., allocate [6, 12, 12]?

Potential Goal: Max-Min Fairness

- The allocation must be “feasible”
 - Total allocation should not exceed link capacity
- Can’t be any better for the ‘min’
 - Any attempt to increase the allocation of one user
 - ... necessarily decreases for another user with equal or lower allocation
- Benefit: Fully utilize a “bottlenecked” resource
 - If demand exceeds capacity, the link is fully used
- How: Progressive filling algorithm
 - Grow all rates until some users stop having demand
 - Continue increasing all remaining rates until link is full

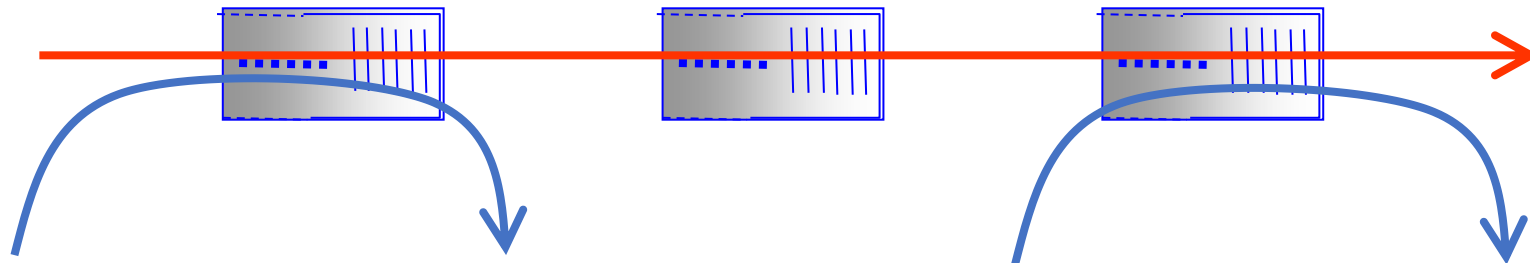
Resource Allocation Over Paths



- Maximum throughput: [30, 30, 0]
 - Total throughput of 60, but user C starves
- Max-min fairness: [15, 15, 15]
 - Equal allocation, but throughput of just 45
- Proportional fairness: [20, 20, 10]
 - Balance trade-off between throughput and equality
 - Throughput of 50, and penalize C for using 2 busy links

TCP Achieves a Notion of Fairness

- Effective utilization is not only goal
 - We also want to be *fair* to various flows
- Simple definition: equal bandwidth shares
 - N flows that each get $1/N$ of the bandwidth?
- But, what if flows traverse different paths?
 - Result: **bandwidth shared in proportion to RTT**



Conclusions

- Congestion is inevitable
 - Internet does not reserve resources in advance
 - TCP actively tries to push the envelope
- Congestion can be handled
 - TCP sender limits traffic to a congestion window
 - Additive increase, multiplicative decrease
- Fairness
 - TCP congestion control is a distributed algorithm that achieves “fairness”
 - ... well, as long as TCP end-points don't cheat!

