Resource Allocation: Congestion Control

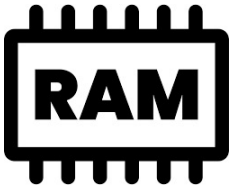
COS 316: Principles of Computer System Design

Lecture 14

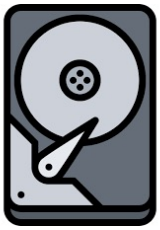
Amit Levy & Jennifer Rexford

1

Systems Have Limited Resources

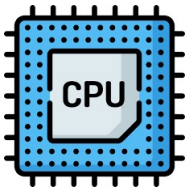


memory



disk

bandwidth



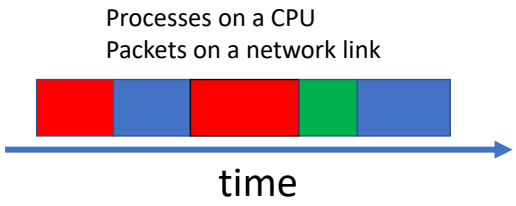
processor

energy

2

Resource Contention and Resource Allocation

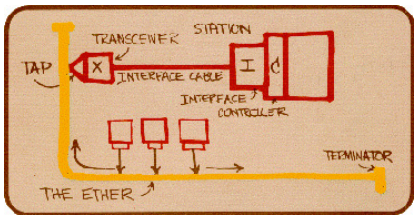
- System resources are shared by multiple users
  - Resource contention: conflict over an oversubscribed resource
  - Resource allocation: assigning resources to various users
- Application process
  - Use of the CPU for processing
  - Use of physical memory for virtual memory
- Network sockets
  - Use of the network bandwidth

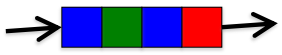


3

Internet Congestion

- Best-effort networks do 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

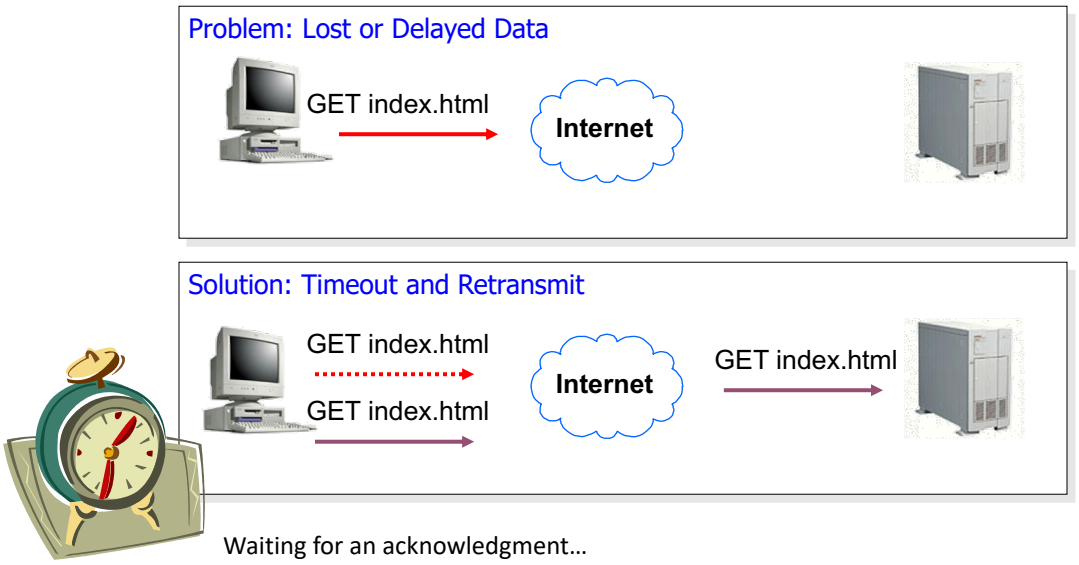




queue

4

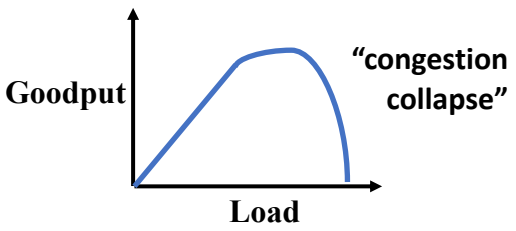
Reminder: TCP Retransmits Lost or Delayed Packets



5

TCP Congestion Collapse

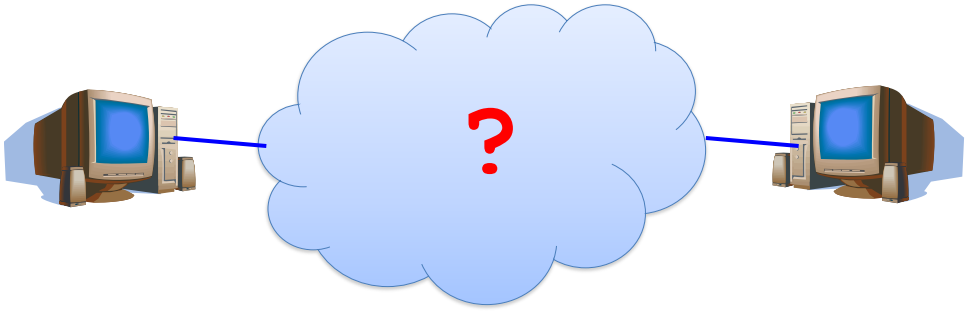
- Easily leads to *congestion collapse*
  - Senders retransmit the lost packets
  - Leading to even *greater* load
  - … and even *more* packet loss



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

6

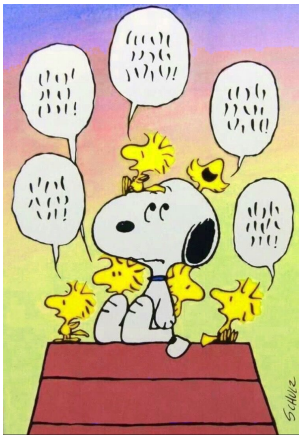
Detect and Respond to Congestion



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

7

Detecting Congestion



- Link layer
  - Carrier sense multiple access
  - Seeing your own frame collide with others
- Network layer
  - Observing end-to-end performance
  - Packet delay or loss over the path

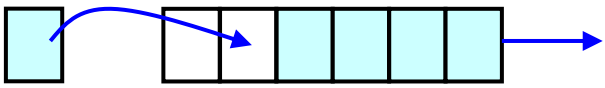
8

TCP Congestion Control

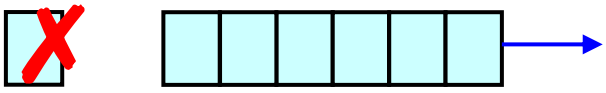
9

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

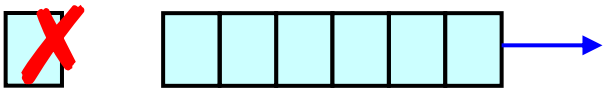


10

How it Looks to the End Host

- Delay: Packet experiences high delay
- Loss: Packet gets dropped along path

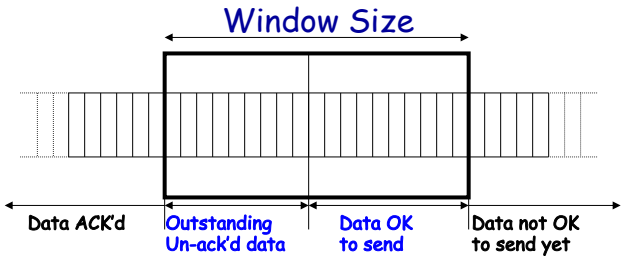
- How does TCP sender learn this?
  - Delay: Round-trip time estimate
  - Loss: Timeout and/or duplicate acknowledgments



11

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

12

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 }

13

13

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

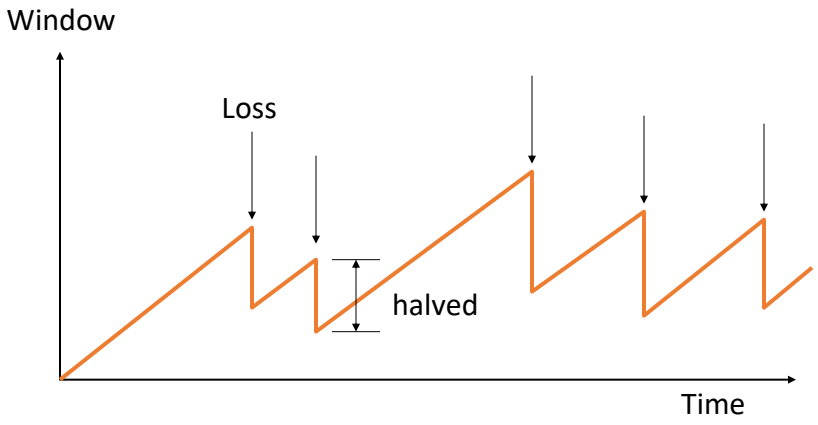
14

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 Max Segment Size (MSS)
- 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)

16

Leads to the TCP "Sawtooth" Behavior



17

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

18

Sources of Poor TCP Performance

- The below conditions *may* primarily result in:

(A) Higher packet latency (B) Greater loss (C) Lower thruput

1. Larger buffers in routers	(A) Higher latency
2. Smaller buffers in routers	(B) Greater Loss
3. Smaller buffers on end-hosts	(C) Lower Throughput
4. Slow application receivers	(C) Lower Throughput

19

TCP Seeks "Fairness"

20

Fair and Efficient Use of a Resource

- Suppose *n* users share a single resource
  - Like the bandwidth on a single link
  - E.g., 3 users sharing a 30 Gbps link
- What is a *fair* allocation of bandwidth?
  - Suppose user demand is "elastic" (i.e., 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]?

21

Fair Use of a Single Resource

- What if some users have *inelastic* 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 he wants?
  - E.g., allocate [6, 12, 12]?

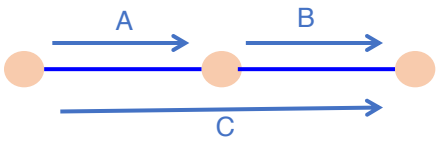
22

Max-Min Fairness

- The allocation must be "feasible"
  - Total allocation should not exceed link capacity
- Protect the less fortunate
  - Any attempt to *increase* the allocation of one user
  - … necessarily *decreases* for another user with equal or lower allocation
- Fully utilize a "bottlenecked" resource
  - If demand exceeds capacity, the link is fully used
- Progressive filling algorithm
  - Grow all rates until some users stop having demand
  - Continue increasing all remaining rates till link is full

23

Resource Allocation Over Paths



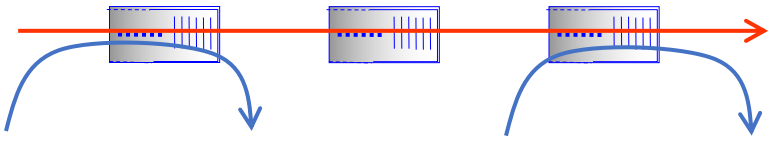
Three users A, B, and C
Two 30 Gbps links

- 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

24

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



25

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 of congestion window size
- Fairness
  - TCP congestion control is a distributed algorithm that achieves “fairness”
  - … well, as long as TCP end-points don’t cheat!

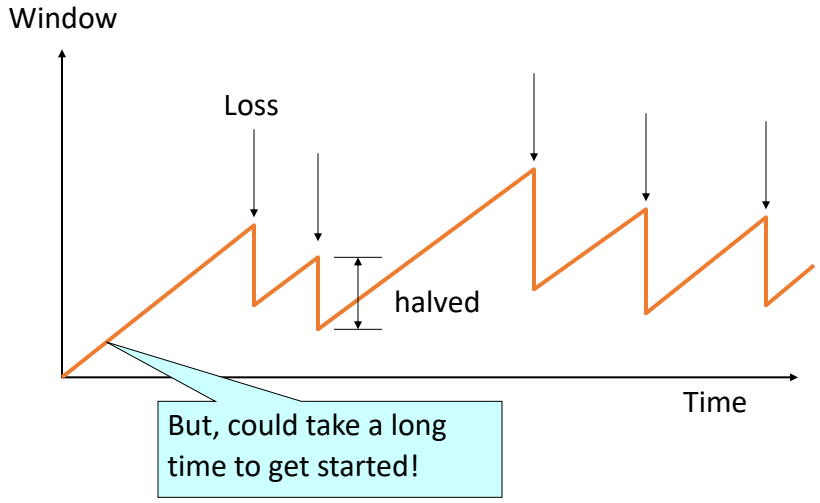
26

Starting a New Flow

27

How Should a New Flow Start?

Start slow (a small CWND) to avoid overloading network



28

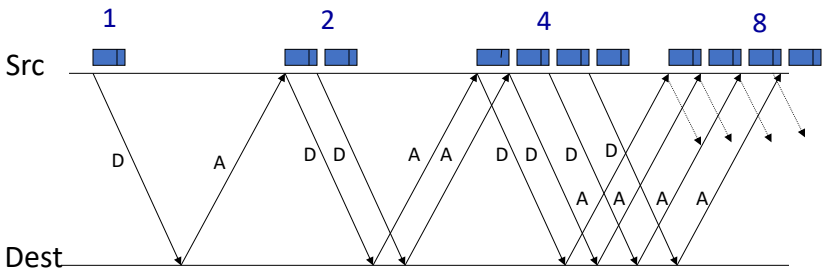
"Slow Start" Phase

- Start with a small congestion window
  - Initially, CWND is 1 MSS
  - So, initial sending rate is MSS / RTT
- Could be pretty wasteful
  - Might be much less than actual bandwidth
  - Linear increase takes a long time to accelerate
- Slow-start phase (really "fast start")
  - Sender starts at a slow rate (hence the name)
  - … but increases rate exponentially until the first loss

29

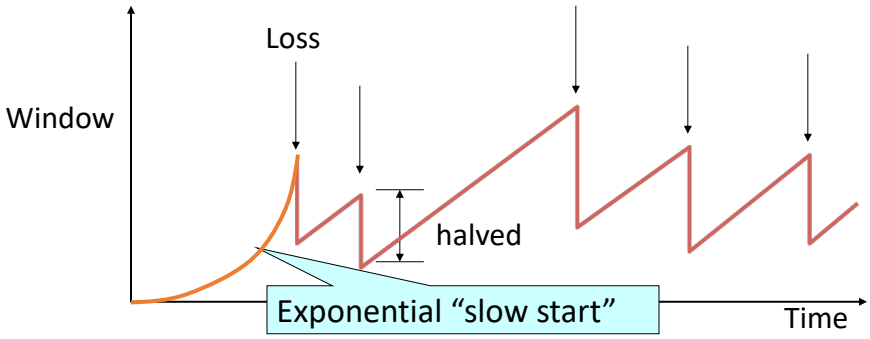
Slow Start in Action

Double CWND per round-trip time



30

Slow Start and the TCP Sawtooth



- TCP originally had *no* congestion control
  - Source would start by sending entire receiver window
  - Led to congestion collapse!
  - "Slow start" is, comparatively, slower

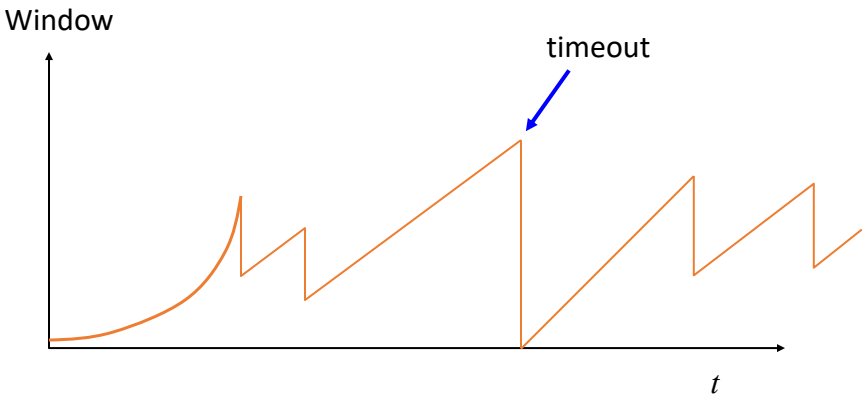
31

Two Kinds of Loss in TCP

- Timeout vs. Triple Duplicate ACK
  - Which suggests network is in worse shape?
- Timeout
  - If entire window was lost, buffers may be full
  - ...blasting entire CWND would cause another burst
  - ...be aggressive: start over with a low CWND
- Triple duplicate ACK
  - Might be do to bit errors, or "micro" congestion
  - ...react less aggressively (halve CWND)

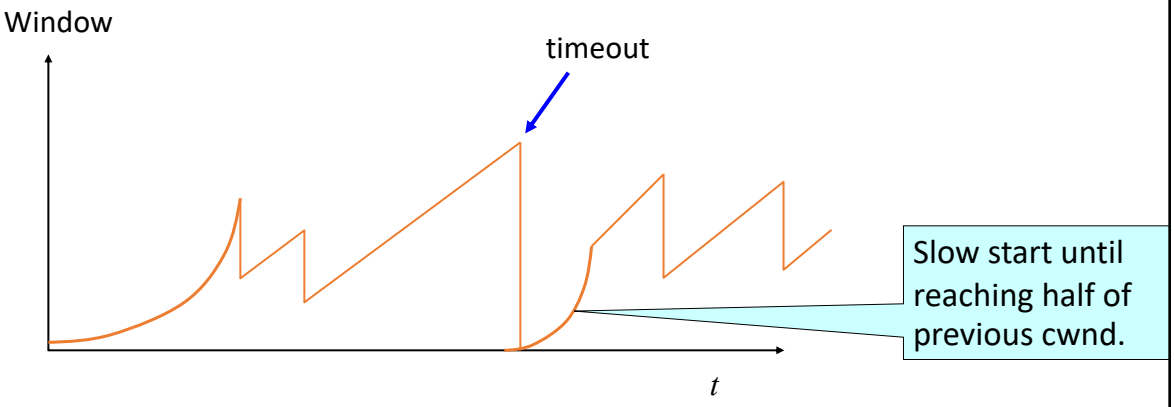
32

Repeating Slow Start After Timeout



33

Repeating Slow Start After Timeout



Slow-start restart: Go back to CWND of 1, but take advantage of knowing the previous value of CWND.

34

Repeating Slow Start After Idle Period

- Suppose a TCP connection goes idle for a while
- Eventually, the network conditions change
  - Maybe many more flows are traversing the link
- Dangerous to start transmitting at the old rate
  - Previously-idle TCP sender might blast network
  - … causing excessive congestion and packet loss
- So, some TCP implementations repeat slow start
  - Slow-start restart after an idle period

35