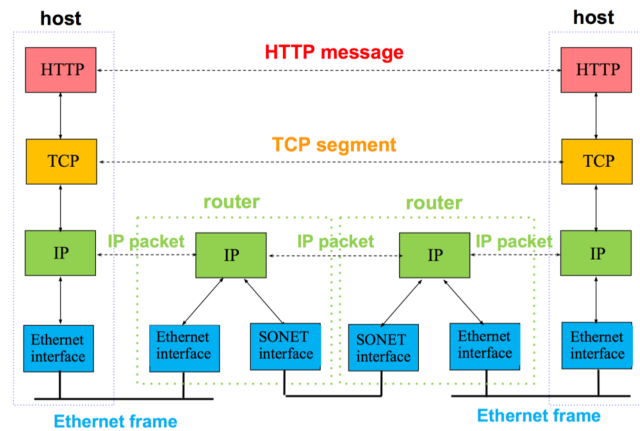
Congestion Control

Mike Freedman
(guest lecture)



2

It's a shared world

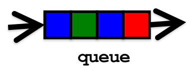
How do we coordinate?

Congestion Control

Distributed Resource Sharing

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

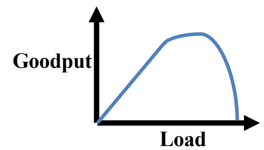


- Excess packets are simply dropped
  - And the sender can simply retransmit

5

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

Responding to Congestion

- Upon detecting congestion
  - Decrease the sending rate
- But, what if conditions change?
  - If more bandwidth becomes available,
  - … unfortunate to keep sending at a low rate
- Upon *not* detecting congestion
  - Increase sending rate, a little at a time
  - See if packets get through

9

Ethernet CSMA/CD

10

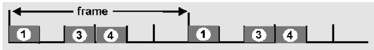
Collisions

- Single shared broadcast channel
  - Avoid having multiple nodes speaking at once
  - Otherwise, collisions lead to garbled data

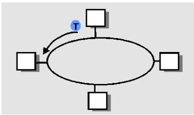
11

Multi-Access Protocol

- Divide channel into pieces
  - In time
  - In frequency



- Take turns
  - Pass a token for the right to transmit



- Punt
  - Let collisions happen
  - Detect and recover from them

12

Multi-Access Protocol

- Divide channel into pieces
  - In time
  - In frequency
- Take turns
  - Pass a token for the right to transmit
- Punt
  - Let collisions happen
  - Detect and recover from them

(a) Efficient/fair at high load, inefficient at low load
(b) Inefficient at high load, efficient/fair at low load

(a) Inefficient at high load
(b) Efficient at all loads
(c) Robust to failures

(a) Inefficient at low load
(b) Efficient at all load
(c) Robust to failures

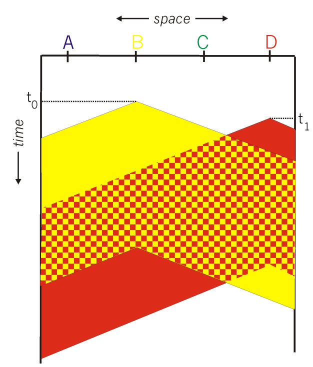
Like Human Conversation…

- Carrier sense
  - Listen before speaking
  - …and don't interrupt!
- Collision detection
  - Detect simultaneous talking
  - … and shut up!
- Random access
  - Wait for a random period of time
  - … before trying to talk again!

Please Wait…

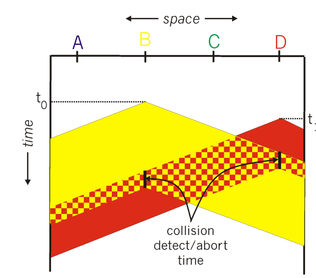
Carrier Sense Multiple Access

- Listen for other senders
  - Then transmit your data
- Collisions can still occur
  - Propagation delay
  - Wasted transmission

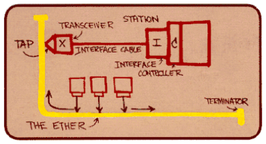


CSMA/CD Collision Detection

- Detect collision
  - Abort transmission
  - Jam the link
- Wait random time
  - Transmit again
- Hard in wireless
  - Must receive data while transmitting



Ethernet Uses CSMA/CD

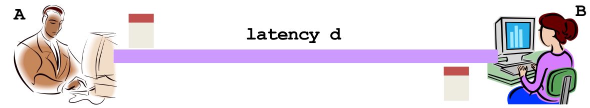


Metcalfe's Ethernet sketch

- Carrier Sense: wait for link to be idle
- Collision Detection: listen while transmitting
- Random Access: exponential back-off
  - After collision, wait random time before trying again
  - After m^{th} collision, choose K randomly from {0, …, 2^m-1}
  - … and wait for K*512 bit times before trying again

17

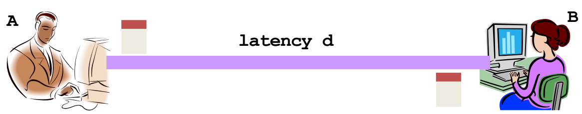
Limitations on Ethernet Length



- Latency depends on physical length of link
  - Time to propagate a packet from one end to other
- Suppose A sends a packet at time t
  - And B sees an idle line at a time just before t+d
  - … so B happily starts transmitting a packet
- B detects a collision, and sends jamming signal
  - But A doesn't see collision till t+2d

18

Limitations on Ethernet Length



- A needs to wait for time 2d to detect collision
  - So, A should keep transmitting during this period
  - … and keep an eye out for a possible collision
- Imposes restrictions on Ethernet
  - Maximum length of the wire: 2500 meters
  - Minimum length of the packet: 512 bits (64 bytes)

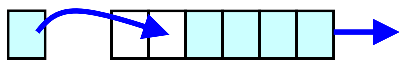
19

TCP Congestion Control

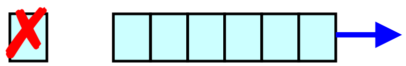
20

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

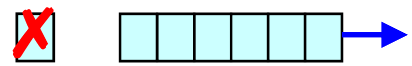


21

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



22

TCP Congestion Window

- Each TCP sender maintains a congestion window
  - Max number of bytes to have in transit (not yet ACK'd)
- Adapting the congestion window
  - Decrease upon losing a packet: backing off
  - Increase upon success: optimistically exploring
  - Always struggling to find right transfer rate
- Tradeoff
  - Pro: avoids needing explicit network feedback
  - Con: continually under- and over-shoots "right" rate

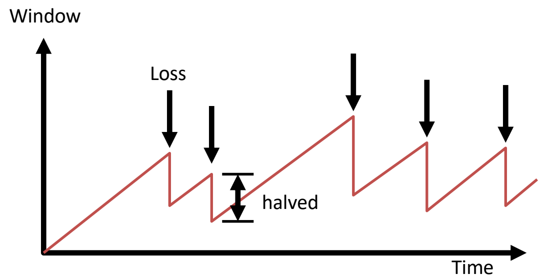
23

Additive Increase, Multiplicative Decrease

- How much to adapt?
  - Additive increase: On success of last window of data, increase window by 1 Max Segment Size (MSS)
  - Multiplicative decrease: On 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 thruput)
  - AIMD: A necessary condition for stability of TCP

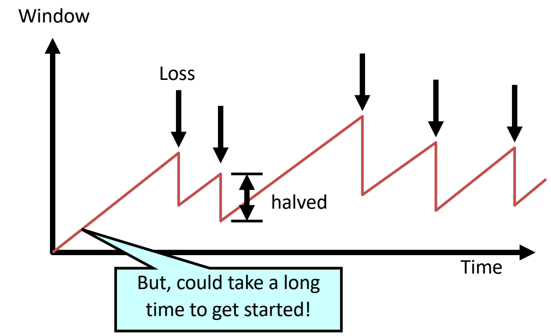
24

Leads to the TCP "Sawtooth"



How Should a New Flow Start?

Start slow (a small CWND) to avoid overloading network

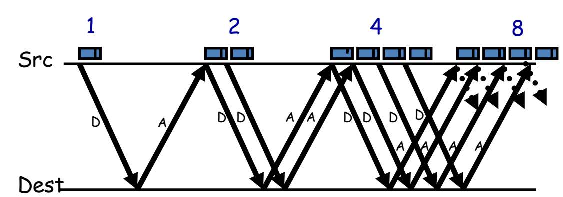


"Slow Start" Phase

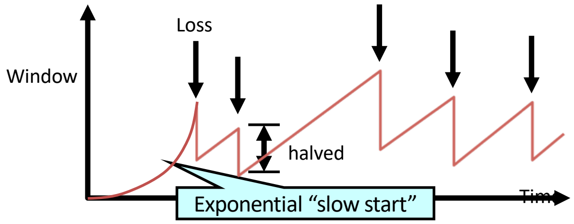
- Start with a small congestion window
  - Initially, CWND is 1 MSS
  - So, initial sending rate is MSS / RTT
- Could be pretty wasteful
  - Might be much less than actual bandwidth
  - Linear increase takes a long time to accelerate
- Slow-start phase (really "fast start")
  - Sender starts at a slow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

Slow Start in Action

Double CWND per round-trip time



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

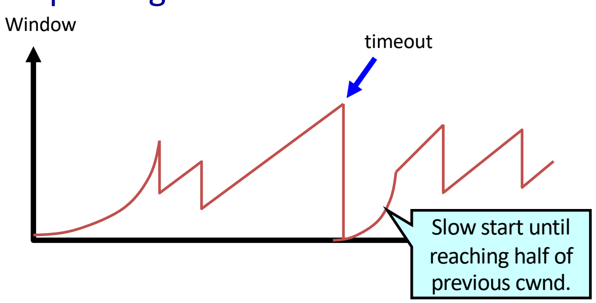
29

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)

30

Repeating Slow Start After Timeout



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

31

Conclusions

- Congestion is inevitable
  - Internet does not reserve resources in advance
  - TCP actively tries to push the envelope
- Congestion can be handled
  - Additive increase, multiplicative decrease
  - **Exponential** backoff: congestion bad, react aggressively
    - Ethernet: *double* retransmission timer
    - TCP: divide sending rate in *half*
- Fundamental tensions
  - Feedback from the network?
  - Enforcement of "TCP friendly" behavior?

32