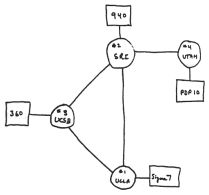
Lecture 3
TCP and Network Architecture



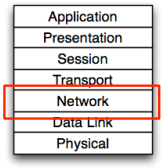
Back to the Network

- The Web uses HTTP, but is built on TCP/IP (like almost all of the Internet)
- Understanding HTTP requires understanding TCP
- No handout on this material yet
- Last networking lecture for awhile

2

Topic 1

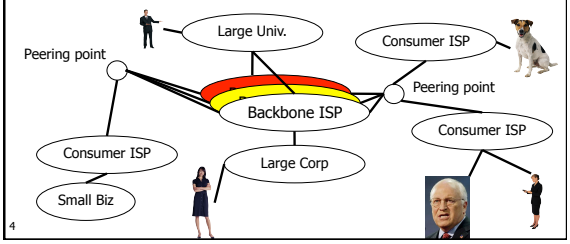
- IP review



3

IP Review

- Best effort packet-switched network
  - Basic unit is *packet*
  - Packets may arrive late, or not at all
  - IP routers form the core of the internet



4

IP Review (2)

- IP addresses are 32 bits
  - E.g., 192.168.1.1
- Forwarding-based networking
  - Each host has a routing table
  - Forward packet to "longest prefix match"

Destination	Gateway
Default	192.168.1.1
192.168.212.111	0:1f:6d:e8:18:0

5

IP Review (3)

- `traceroute` uses debug messages to expose packet's route to destination

```
traceroute to google.com (74.125.95.99), 64 hops max, 52 byte packets
 1  141.212.111.1 (141.212.111.1)  1.037 ms  0.681 ms  1.020 ms (Ann Arbor, MI)
 2  13-cesnbin-arb.r-bin-arbl.umnet.umich.edu (192.12.80.177)  0.566 ms  0.378 ms  0.328 ms
 3  13-barb-bseb-2.r-bin-seb.umnet.umich.edu (192.12.80.11)  0.464 ms  0.495 ms  0.485 ms
 4  (Ann Arbor, MI)
 5  207.72.112.46 (207.72.112.46)  6.552 ms  6.609 ms  6.495 ms(Ann Arbor, MI)
 6  216.239.48.154 (216.239.48.154)  6.785 ms  52.824 ms  6.722 ms(Mountain View, CA)
 7  209.85.241.29 (209.85.241.29)  17.603 ms(Mountain View, CA)
 8  72.14.239.193 (72.14.239.193)  18.306 ms  25.136 ms(Mountain View, CA)
 9  209.85.241.22 (209.85.241.22)  13.167 ms (Mountain View, CA)
```

6

IP Review (4)

- No reason the links have to make sense
- `traceroute urbanspoon.com`
  - Ann Arbor, MI
  - San Mateo, CA
  - New York, NY
  - <ends>

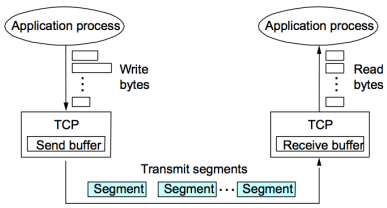
7

TCP

- Transport Control Protocol
  - Reliable, ordered bytestreams
  - Connection-oriented, unlike IP
  - "Virtual circuit" networking built on packet infrastructure
- Processes tied to "host:port" pairs
  - Ports are an OS-level concept
  - Server ports are "well-known" and associated with services
  - Other ports are "ephemeral"

8

TCP Delivery



9

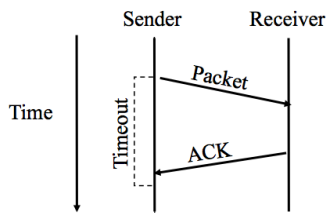
Reliable connections

- Basic principle of reliable TCP connection is retransmission
  - Each packet has 32-bit Sequence Number
  - Every SeqNo is ACKed by receiver
  - When timeout expires, sender emits again
  - Simple!
- OK, maybe not quite that easy

10

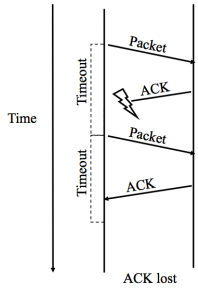
Stop and Wait

- Send a packet, wait for ACK
- Receiver ACKs everything



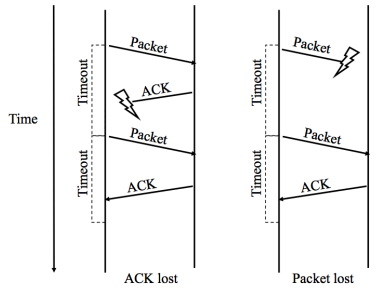
11

Recovering from errors



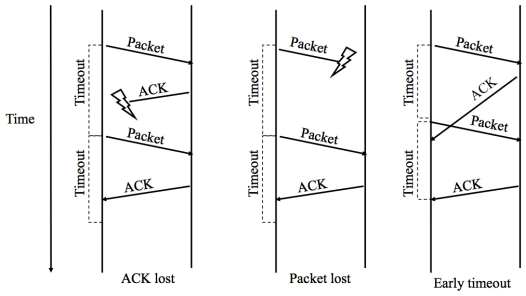
12

Recovering from errors



13

Recovering from errors



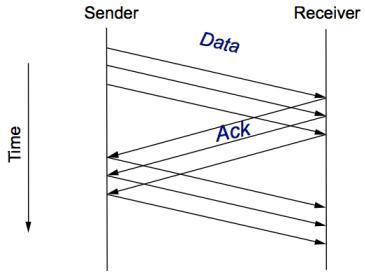
14

Flow Control

- Any downside to Stop-and-Wait?
- **Sliding window** technique for managing send/receive capacity
  - Receiver indicates "receive window" it is willing to buffer
  - Sender cannot have more packets in transit (unACKed) than what can fit in the window
  - Ideally, window is bandwidth * RT delay
    - Keeps as much data in transit as possible

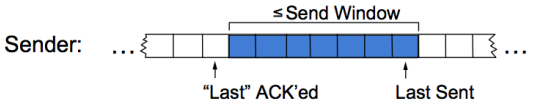
15

Sliding Window Ideal



16

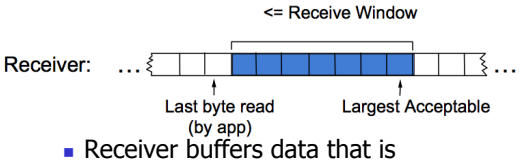
Sliding Window - Sender



- Sender buffers unACKed data
- ACKs are always in-order

17

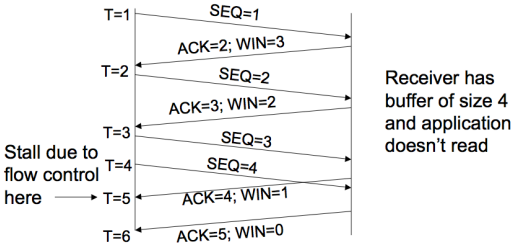
Sliding Window - Receiver



- Receiver buffers data that is
  - Out-of-order
  - Not yet read off by application
- ACKs data as it arrives
- Also shrinks/expands advertised window in response to application behavior

18

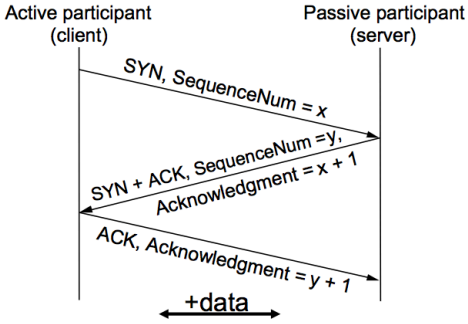
Sliding Window IN ACTION



19

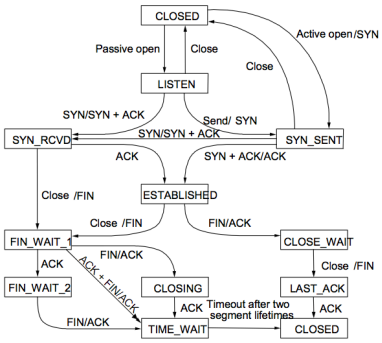
TCP Connection Setup

- The World Famous 3-way Handshake



20

Transitions



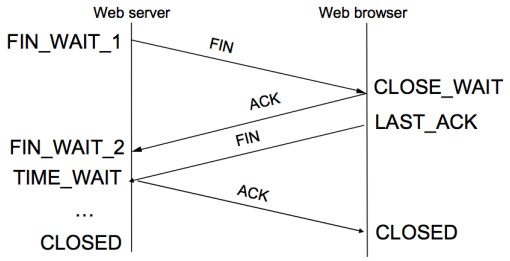
21

Connection Teardown

- Orderly release by sender + receiver
  - "Hanging up the phone"
- Releases state on both sides

22

TCP Connection Teardown



23

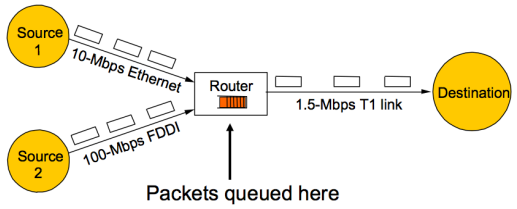
TIME_WAIT

- We wait 2MSL (twice the max segment lifetime) before completing the close
  - ACK could have been lost, so FIN will be resent
  - Could interfere with subsequent connection

24

Congestion Control

- Buffers in middle of network can become overloaded
- Not part of window negotiation



25

Congestion Control

- When has packet been lost?
  - Too long a timer, you're waiting pointlessly
  - Too short, you're adding needless load
- Many retransmits can induce *congestion collapse* (it happened in late 1980s!)
  - Retransmits just add to congestion
  - Capacity of network falls dramatically
- **Congestion control** is the art of dynamically sizing the sender's window to avoid slamming in-network routers
  - Ideally based on round-trip time

26

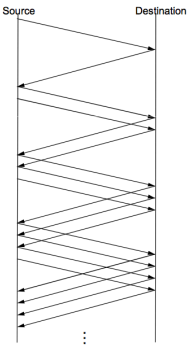
Probing

- Hosts use *packet-probing* to figure out bandwidth to remote side
- A lost packet is assumed to be dropped due to router buffer congestion
  - Good assumption on internet
  - What about on wireless?

27

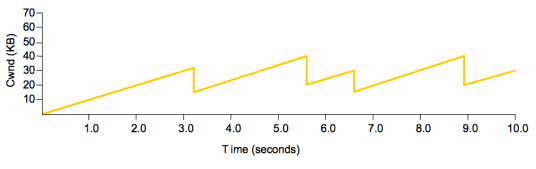
One approach: AIMD

- Additive Increase, Multiplicative Decrease
- Increase slowly if there is bandwidth
- Back down fast in event of loss



28

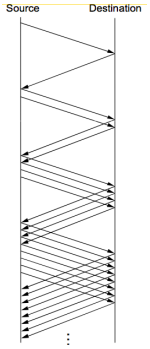
TCP sawtooth



29

Another: Slow Start

- AIMD can take ages to find best level
- Instead, SS *doubles* every RTT
- Use Slow Start at first; then AIMD at SS threshold



30

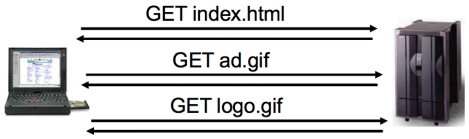
HTTP on TCP

- TCP Summary
  - A roundtrip for each window of bytes
  - Takes some time to find best rate
  - OS buffer overhead for each connection
- HTTP Summary
  - Connections short-lived
  - Small amounts of data per connection
  - Many connections per HTML page
- Naïve HTTP on TCP yields *awful* performance

31

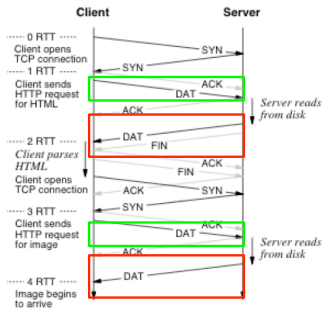
HTTP 1.0 on TCP

- Every HTML-embedded item requires a GET



32

HTTP on TCP



33

HTTP/1.1

- Persistent Conections
  - One TCP connection, many HTTP methods
    - Good for small objects, not large
    - Makes parallel downloads difficult

34

Persistent HTTP Performance

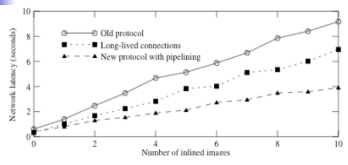


Image size=2544

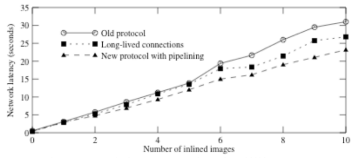


Image size=45566

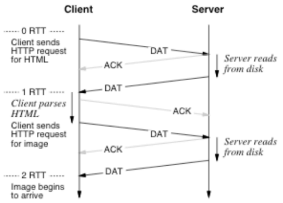
35

HTTP/1.1

- Persistent Conections
  - One TCP connection, many HTTP methods
    - Good for small objects, not large
    - Makes parallel downloads difficult
- Caching
  - GET + IF-MODIFIED-SINCE <timestamp>
  - When combined with browser cache, eliminates a lot of unneeded data transfer
  - Same number of roundtrips
  - Lots of different caches possible
    - Browser, department proxy, Akamai

36

HTTP/1.1


Client
Server
0 RTT
Client sends HTTP request for HTML
DAT
ACK
Server reads from disk
1 RTT
Client parses HTML
Client sends HTTP request for image
DAT
ACK
Server reads from disk
2 RTT
Image begins to arrive
DAT

37