# 15-441 15-641 Computer Networking

TCP Performance and the Web
Peter Steenkiste

Fall 2015
www.cs.cmu.edu/~prs/15-441-F15

## Outline TCP

- TCP status and evolution (last week)
- TCP performance model (today)
  - Throughput model
  - Implications

2

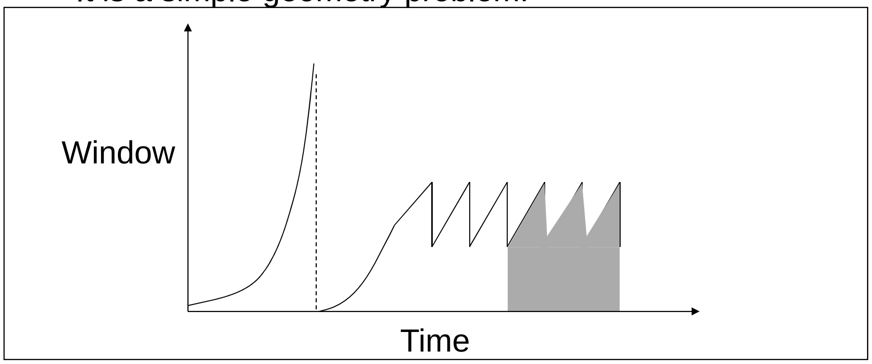
## TCP Modeling

- Given the congestion behavior of TCP can we predict what type of performance we should get?
- What are the important factors
  - Loss rate: Affects how often window is reduced
  - RTT: Affects increase rate and relates BW to window
  - RTO: Affects performance during loss recovery
  - MSS: Affects increase rate (additive increase)

3

## Overall TCP Behavior

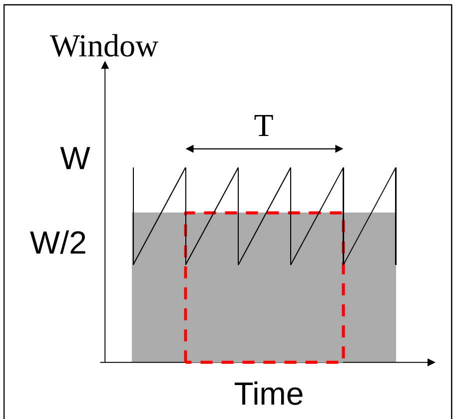
- Let us concentrate on steady state behavior with no timeouts and perfect loss recovery
- Packets transferred = area under curve
  - It is a simple geometry problem!




4

## Transmission Rate

- What is area under curve?
  - Window in packets
  - Time in RTTs
  - A = avg window * time
    = ¾ W * T (packets)
- What was bandwidth?
  - BW = A / T = ¾ W
    - In packets per RTT
  - Convert to bytes per second
  - BW = ¾ W * MSS / RTT
- What is W?
  - Depends on loss rate




5

## Simple TCP Model

- Some additional assumptions
  - Fixed RTT
  - No delayed ACKs
- In steady state, TCP loses a packet each time window reaches W packets
  - Window drops to W/2 packets
  - Each RTT window increases by 1 packet
    → W/2 * RTT between packet losses

6

## Simple Loss Model

- What was the loss rate?
  - Packets transferred = (¾ W/RTT) * (W/2 * RTT) = $3W^2/8$
  - 1 packet lost → loss rate = p = $8/3W^2$
  - $W = \sqrt{\frac{8}{3p}}$
- BW = ¾ * W * MSS / RTT
  - $W = \sqrt{\frac{8}{3p}} = \frac{4}{3} \times \sqrt{\frac{3}{2p}}$
  - $BW = \frac{MSS}{RTT \times \sqrt{2p/3}}$

7

## Implication 1: Fairness

- Flows sharing bottleneck do NOT get same bandwidth!
  - Fairness: BW proportional to 1/RTT
- TCP is "RTT fair"
  - Only "if all else is equal" do flows sharing a bottleneck get the same bandwidth
  - Not by design

8

## Implication 2: High Speed Networks

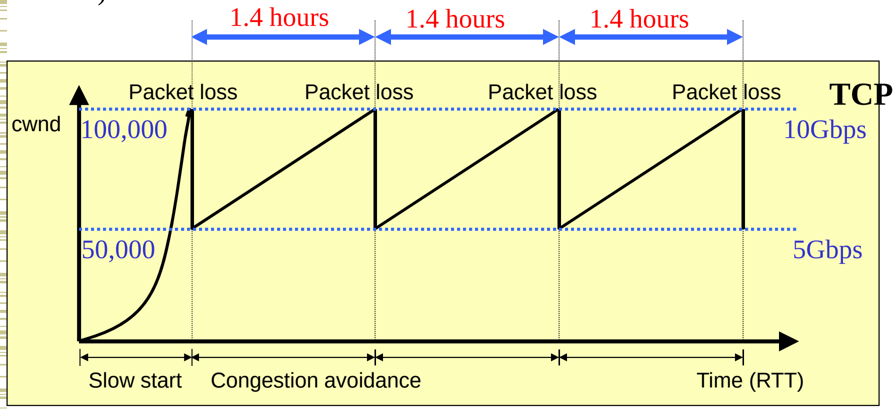
$$T \approx \frac{\sqrt{1.5}\, MSS}{RTT\,\sqrt{p}}$$

- Suppose RTT = 100 ms, MSS = 1.5 KB
- T = 100 Gb/sec
- p=?
  - $p \approx 2 \times 10^{-12}$
- 1 drop every 6 petabits (17 hours).
- So….

9

## TCP over High-Speed Networks

- A TCP connection with 1250-Byte packet size and 100ms RTT is running over a 10Gbps link (assuming no other connections, and no buffers at routers)




Source: Rhee, Xu. "Congestion Control on High-Speed Networks"

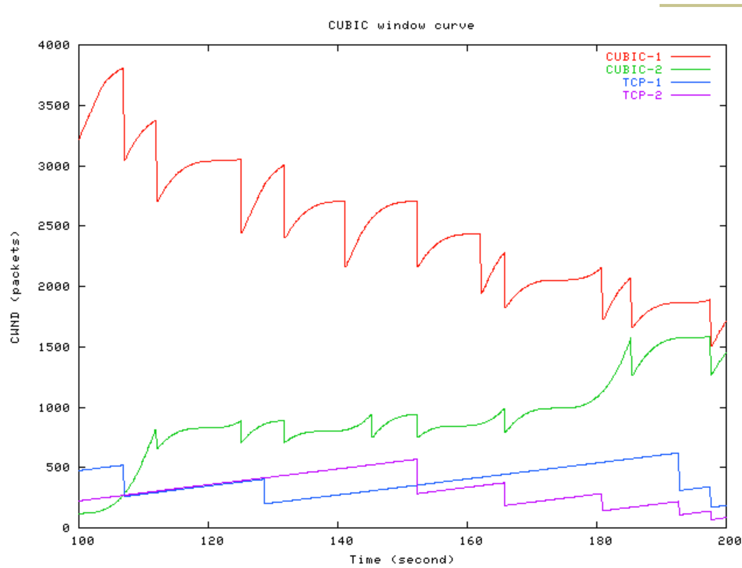
10

# TCP (CU)BIC

- Goal is to spend more time at the high end of the window value range
  - Remember: 1.4 hours to reach Wmax on 10 Gbs link?
- Idea: make the additive increase adaptive depending on how close you are to presumed Wmax value
  - Fast recovery using larger additive increase toward Wmax
  - Slow change around Wmax
  - Fast search for a higher Wmax

11

# TCP CUBIC in one slide




12

## Implication 3: How about non-TCP flow?

- Certain types of flow cannot use TCP
  - E.g., multi-media streaming: timeouts add excessive delays, reducing “Quality of Experience”
  - Require custom transport, but what congestion control?
- Solution TCP: make them “TCP friendly”
  - “Like TCP but smoother”
  - Motivation is that they need to coexist nicely with TCP
  - Should hold their own without clobbering TCP flows
- Their throughput must follow that of the TCP equation
  - Calculated smoothed estimates of RTT, p, …
  - Do TCP-Friendly Rate Control (TFRC) based on TCP equation
  - Adjust rate w/o timeouts



## Overview Content Delivery

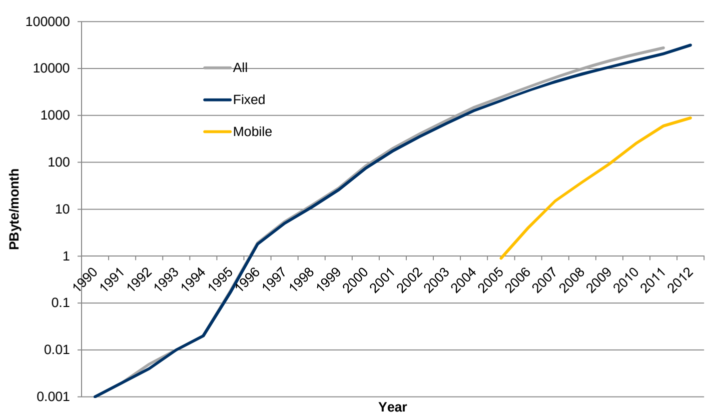
- Web
  - Protocol interactions
  - Caching
  - Cookies
- Peer-to-peer
- CDNs
- Video

## Web history

- 1945: Vannevar Bush, “As we may think”, Atlantic Monthly, July, 1945.
  - Describes the idea of a distributed hypertext system.
  - A “memex” that mimics the “web of trails” in our minds.
- 1989: Tim Berners-Lee (CERN) writes internal proposal to develop a distributed hypertext system
  - Connects “a web of notes with links”.
  - Intended to help CERN physicists in large projects share and manage information
- 1990: TBL writes graphical browser for Next machines
- 1992-1994: NCSA/Mosaic/Netscape browser release

15

## Internet Traffic History




16

## Typical Workload (Web Pages)

- Multiple (typically small) objects per page
- File sizes
  - Heavy-tailed
    - Pareto distribution for tail
    - Lognormal for body of distribution
- Embedded references
  - Number of embedded objects also Pareto
    $Pr(X>x) = (x/x_m)^{-k}$
- This plays havoc with performance. Why?
- Solutions?

- Lots of small objects versus TCP
  - 3-way handshake
  - Lots of slow starts
  - Extra connection state

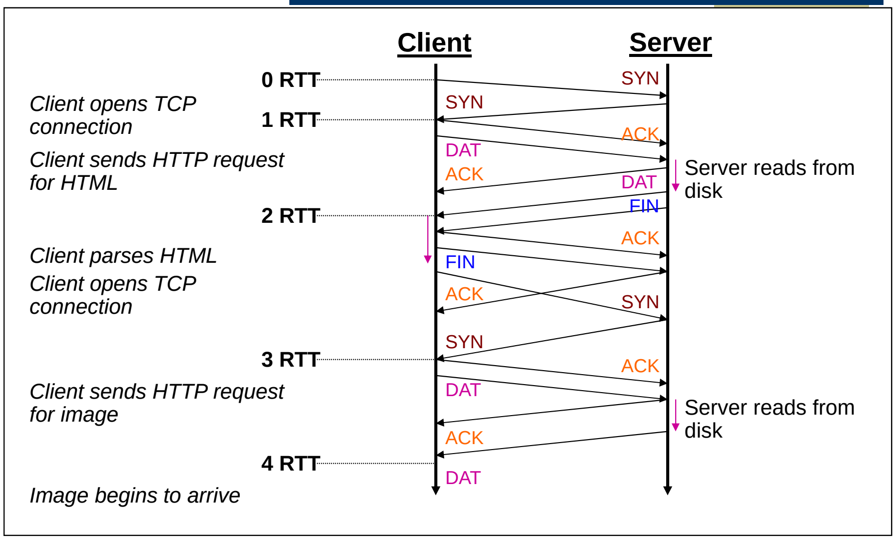
17

## HTTP 0.9/1.0

- One request/response per TCP connection
  - Simple to implement
- Short transfers are very hard on TCP
  - Multiple connection setups → three-way handshake each time
    - Several extra round trips added to transfer
  - Many slow starts – low throughput because of small window
    - Never leave slow start for short transfers
  - Loss recovery is poor when windows are small
  - Lots of extra connections
    - Increases server state/processing

18

## Single Transfer Example

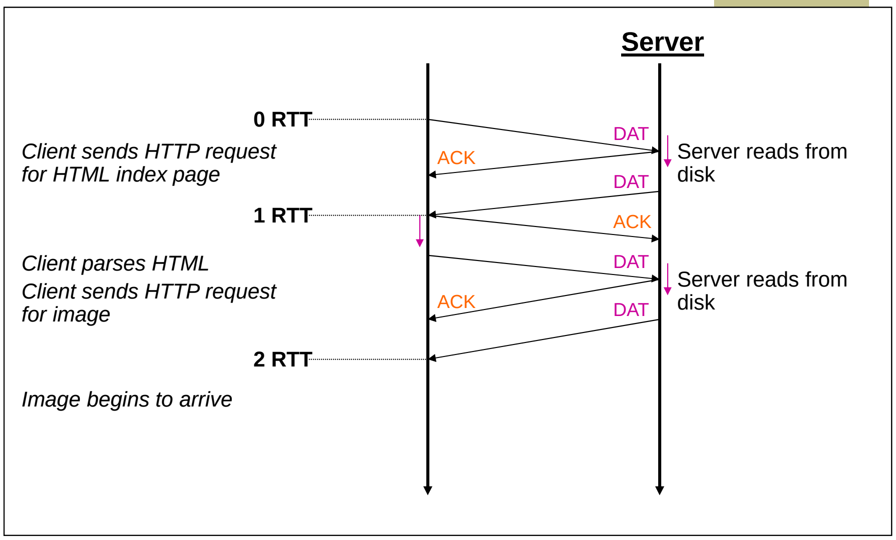





## HTTP 1.1

- Multiplex multiple transfers onto one TCP connection
  - Avoid handshake and slow start when getting multiple objects from same server
- Transfers can be pipelined, i.e., multiple outstanding requests
  - Requests are handled in FIFO order by server
- How to identify requests/responses?
  - Delimiter → Server must examine response for delimiter string
  - Content-length and delimiter → Must know size of transfer in advance
  - Block-based transmission → send in multiple length delimited blocks
  - Store-and-forward → wait for entire response and then use content-length

# Persistent Connection Solution






# Some Challenges with HTTP 1.1

- Head of line blocking: “slow” objects can delay all requests that follow
  - E.g., objects from disk versus objects in cache
- Browsers open multiple TCP connections to achieve parallel transfers
  - Increases load on servers and network
- HTTP headers are big
  - Cost higher for small objects
- Embedded objects add RTT
  - With small objects, RTT dominates

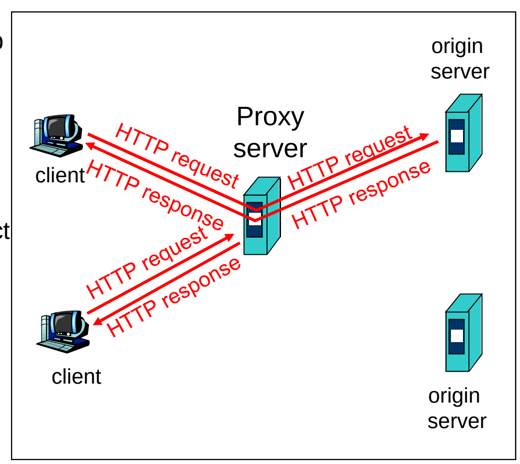
## HTTP 2.0 to the Rescue

- Can multiplex many requests over a TCP connection AND
  - Responses are carried over flow controlled streams – avoids HOL blocking
  - Streams can be prioritized by client based on how critical they are to rendering
- HTTP headers are compressed
- A PUSH features allows server to push embedded objects to the client without waiting for a client request
  - Avoids an RTT
  - What is the challenge?
- Default is to use TLS – fall back on 1.1 otherwise

23

## Web Proxy Caches

- User configures browser: Web accesses via cache
- Browser sends all HTTP requests to cache
  - Object in cache: cache returns object
  - Else cache requests object from origin server, then returns object to client




24

# No Caching Example (1)

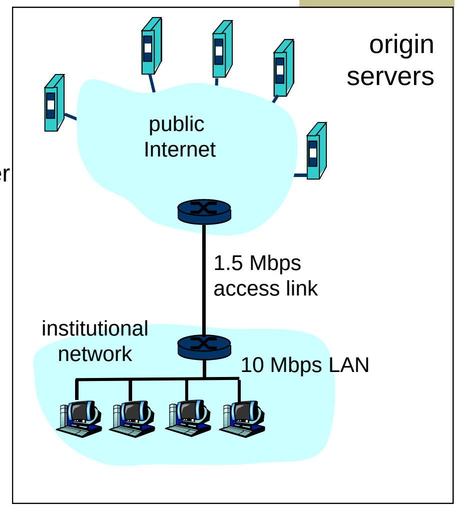
Assumptions

- Average object size = 100,000 bits
- Avg. request rate from institution's browser to origin servers = 15/sec
- Delay from institutional router to any origin server and back to router = 2 sec

Consequences

- Utilization on LAN = 15%
- Utilization on access link = 100%
- Total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds




25

# No Caching Example (2)

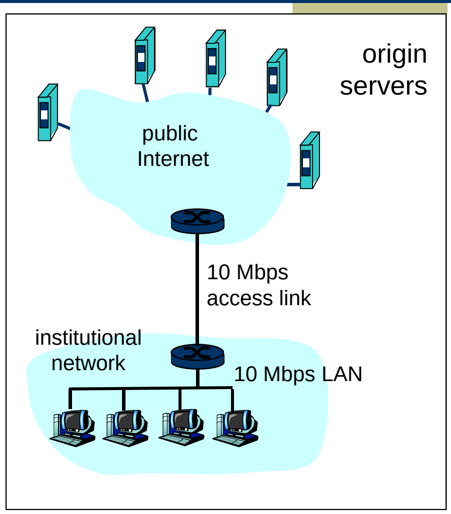
Possible solution

- Increase bandwidth of access link to, say, 10 Mbps
- Often a costly upgrade

Consequences

- Utilization on LAN = 15%
- Utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

= 2 sec + msecs + msecs




26

## With Caching Example (3)

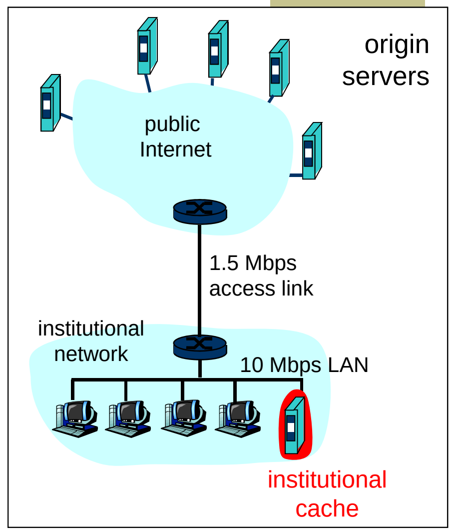
Install cache

- Suppose hit rate is .4

Consequence

- 40% requests will be satisfied almost immediately (say 10 msec)
- 60% requests satisfied by origin server
- Utilization of access link reduced to 60%, resulting in negligible delays
- Weighted average of delays

= .6*2 sec + .4*10msecs < 1.3 secs




27

## HTTP Caching

- Clients often cache documents
  - Challenge: update of documents
  - If-Modified-Since requests to check
    - HTTP 0.9/1.0 used just date
    - HTTP 1.1 has an opaque “entity tag” (could be a file signature, etc.) as well
- When/how often should the original be checked for changes?
  - Check every time?
  - Check each session? Day? Etc?
  - Use Expires header
    - If no Expires, often use Last-Modified as estimate

28

## Problems

- Fraction of HTTP objects that are cacheable is dropping
  - Why?
  - Major exception?
- This problem will not go away
  - Dynamic data → stock prices, scores, web cams
  - CGI scripts → results based on passed parameters
- Other less obvious examples
  - SSL → encrypted data is not cacheable
    - Most web clients don't handle mixed pages well →many generic objects transferred with SSL
  - Cookies → results may be based on past data
  - Hit metering → owner wants to measure # of hits for revenue, etc.
- What will be the end result?

29

## Cookies: Keeping "state"

Many major Web sites use cookies

Four components:

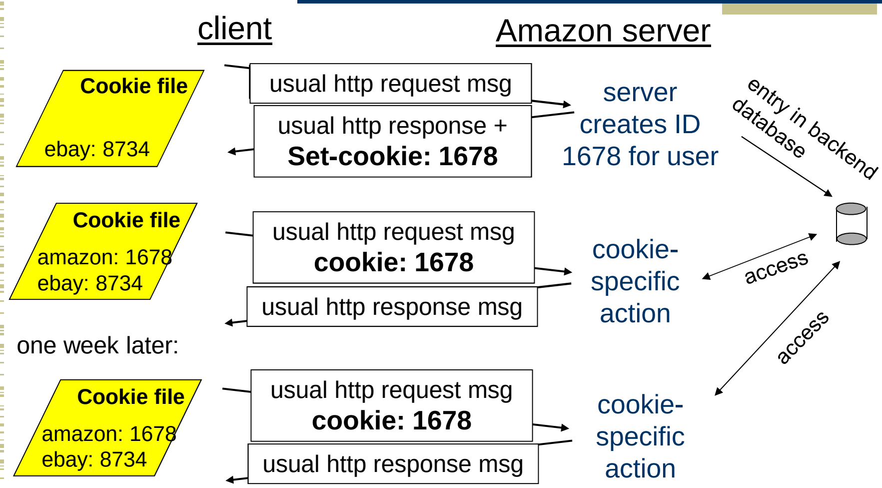
1) Cookie header line in the HTTP response message
2) Cookie header line in HTTP request message
3) Cookie file kept on user's host and managed by user's browser
4) Back-end database at Web site

Example:

- Susan accesses Internet always from the same PC
- She visits a specific e-commerce site for the first time
- When initial HTTP requests arrives at the site, the site creates a unique ID and creates an entry in a backend database for that ID

30

## Cookies: Keeping “State”




31

## Overview Content Delivery

- Web
  - Protocol interactions
  - Caching
  - Cookies
- Peer-to-peer
- CDNs
- Video

32

## Overview Content Delivery

- Web
  - Protocol interactions
  - Caching
  - Cookies
- Peer-to-peer
- CDNs
- Video

33