15-441 15-641 Computer Networking

Lecture 15: The Web
Peter Steenkiste

Fall 2016
www.cs.cmu.edu/~prs/15-441-F16

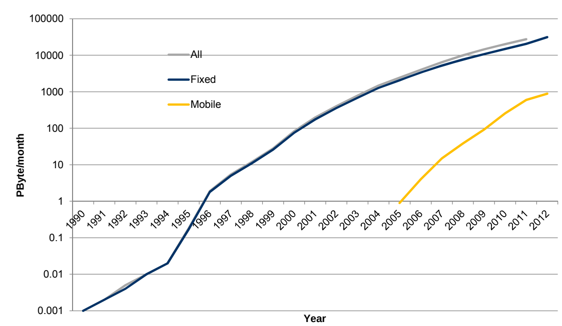
Overview Content Delivery

- Web
  - Protocol interactions
  - HTTP versions
  - 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

Internet Traffic History



Typical Workload (Web Pages)

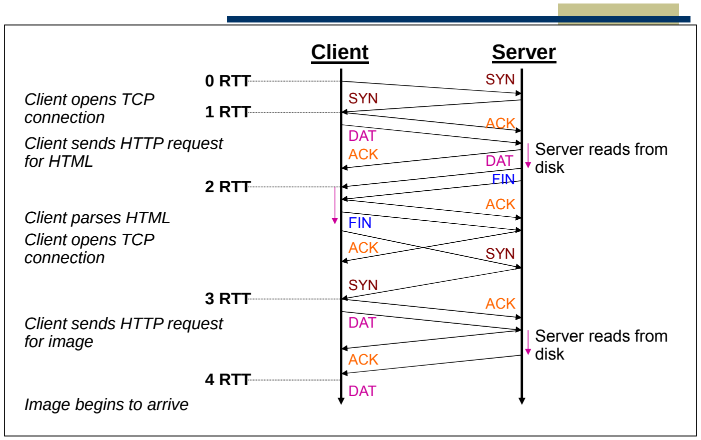
- Multiple (typically small) objects per page
- File sizes
  - Heavy-tailed
    - Pareto distribution for tail
    - Lognormal for body of distribution
- Embedded references
  - Number of embedded objects also Pareto
    $Pr(X>x) = (x/x_m)^{-k}$
- This plays havoc with performance. Why?
- Solutions?

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

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 increase server state and processing overhead

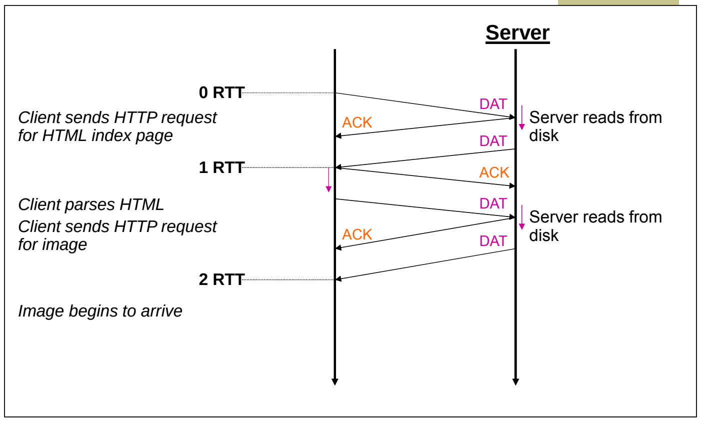
Single Transfer Example



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

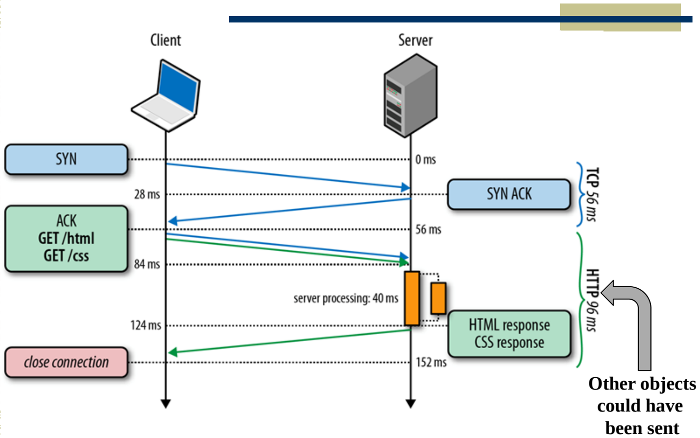
Persistent Connection Solution



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
  - Single “slow” object can delay many “fast” objects
- 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

Example of Head of Line Blocking

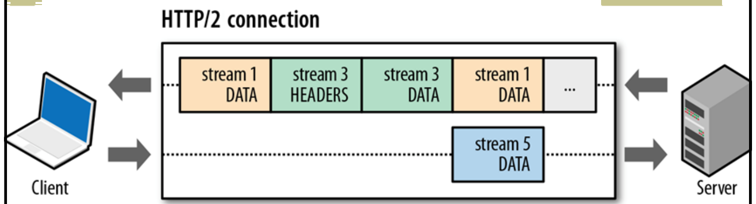


Source: http://chimera.labs.oreilly.com/books/1230000000545/ch11.html

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
  - ≈ multiple prioritized parallel TCP streams
  - Also: fewer handshakes and more traffic (help congestion control)
- 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

HTTP/2 Multi-Streams Multiplexing

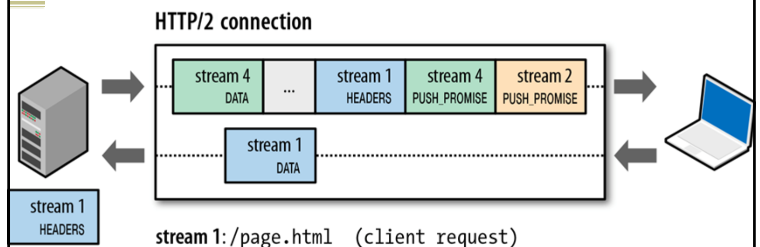


Bit	+0..7	+8..15	+16..23	+24..31
0	Length			Type
32	Flags			
40	R Stream Identifier			
...	*Frame Payload*			

HTTP/2 Binary Framing

https://tools.ietf.org/html/rfc7540

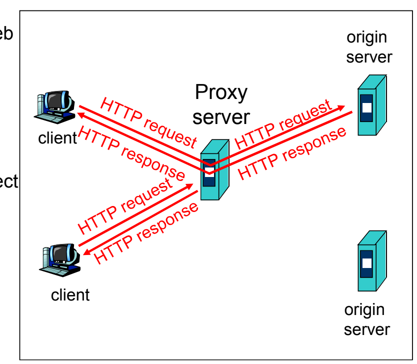
HTTP/2 Server Push



stream 1: `/page.html (client request)`
stream 2: `/script.js (push promise)`
stream 4: `/style.css (push promise)`

Web Proxy Caches

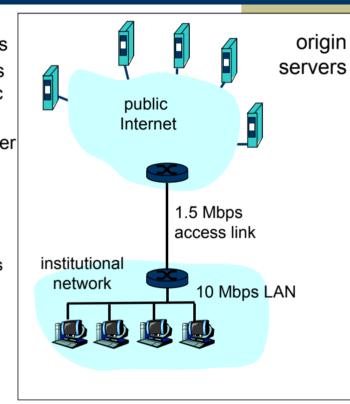
- User configures browser: Web accesses via cache
- Browser sends all HTTP requests to cache
  - Object in cache: cache returns object
  - Else cache requests object from origin server, then returns object to client



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



No Caching Example (2)

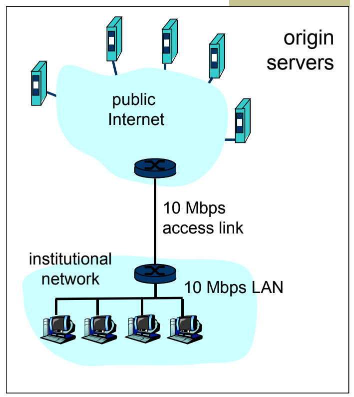
Possible solution

- 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



With Caching Example (3)

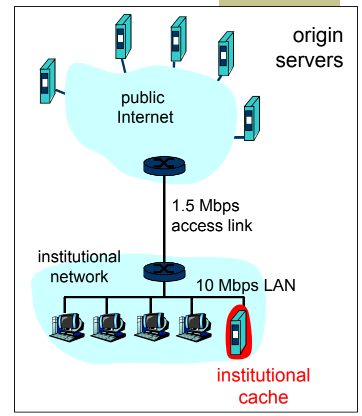
Install cache

- 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



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

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?

Cookies: Keeping “state”

Many major Web sites use cookies

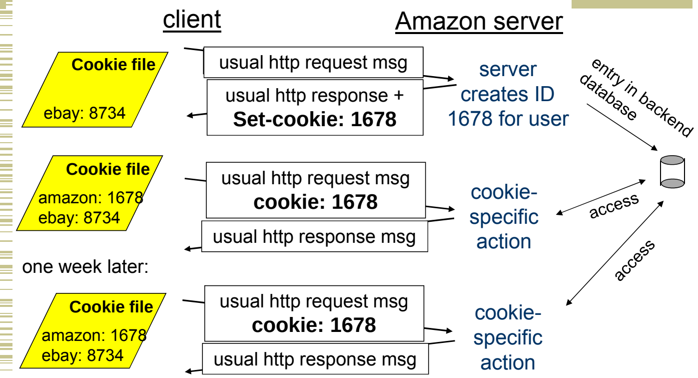
Four components:

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

Cookies: Keeping “State”



Overview Content Delivery

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

Overview Content Delivery

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