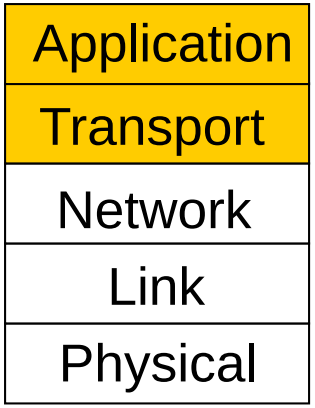
CSE561 – Web Protocols

David Wetherall
djw@cs.washington.edu

Web protocols

- Focus:
  - Applications and their transport needs
- HTTP as an example



Transports we have

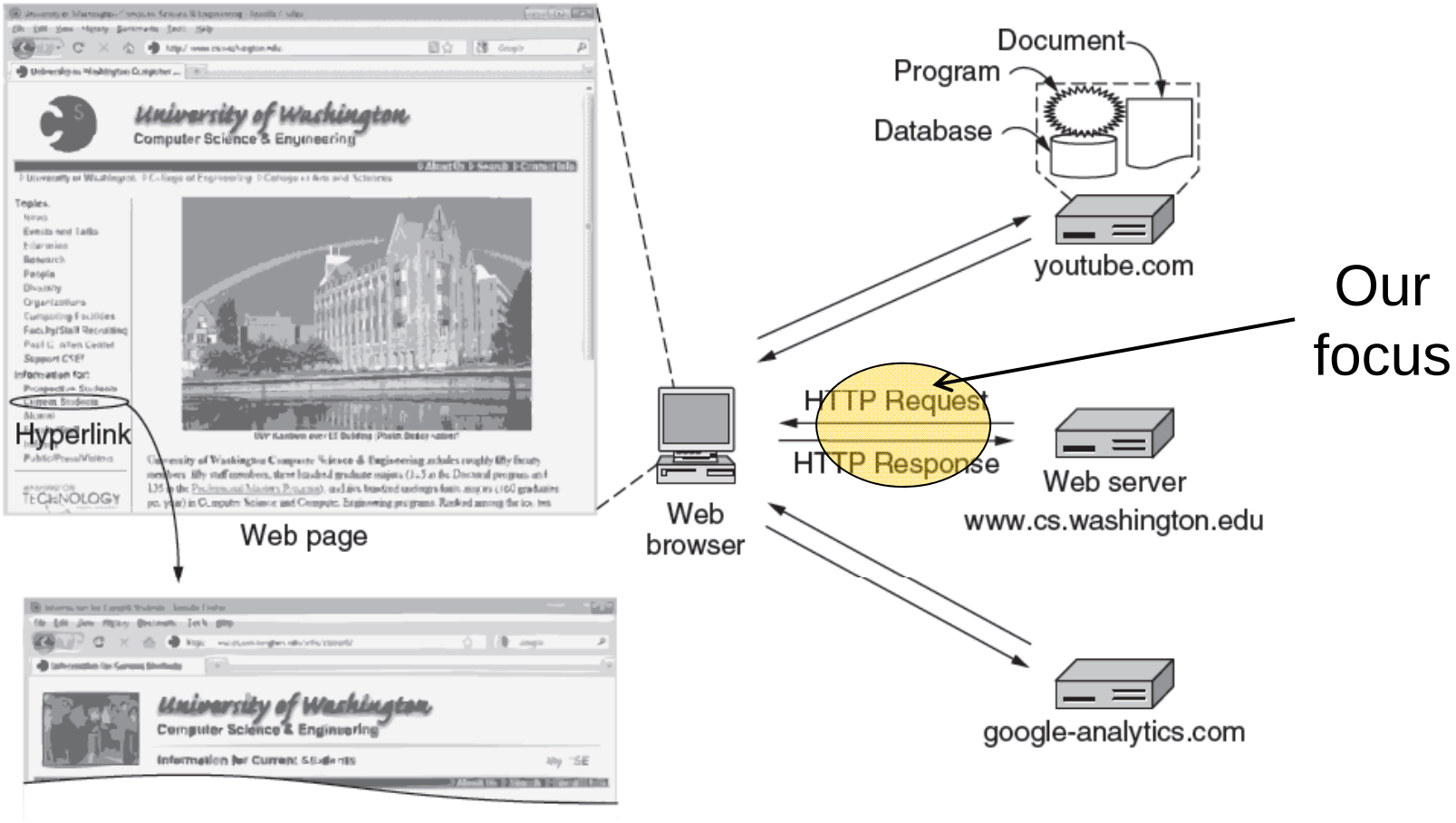
- TCP
  - Reliable, congestion controlled bytestream
- UDP
  - Unreliable individual short messages
  - Error detection if you are nice
  - (Packets!)

Example applications and their needs

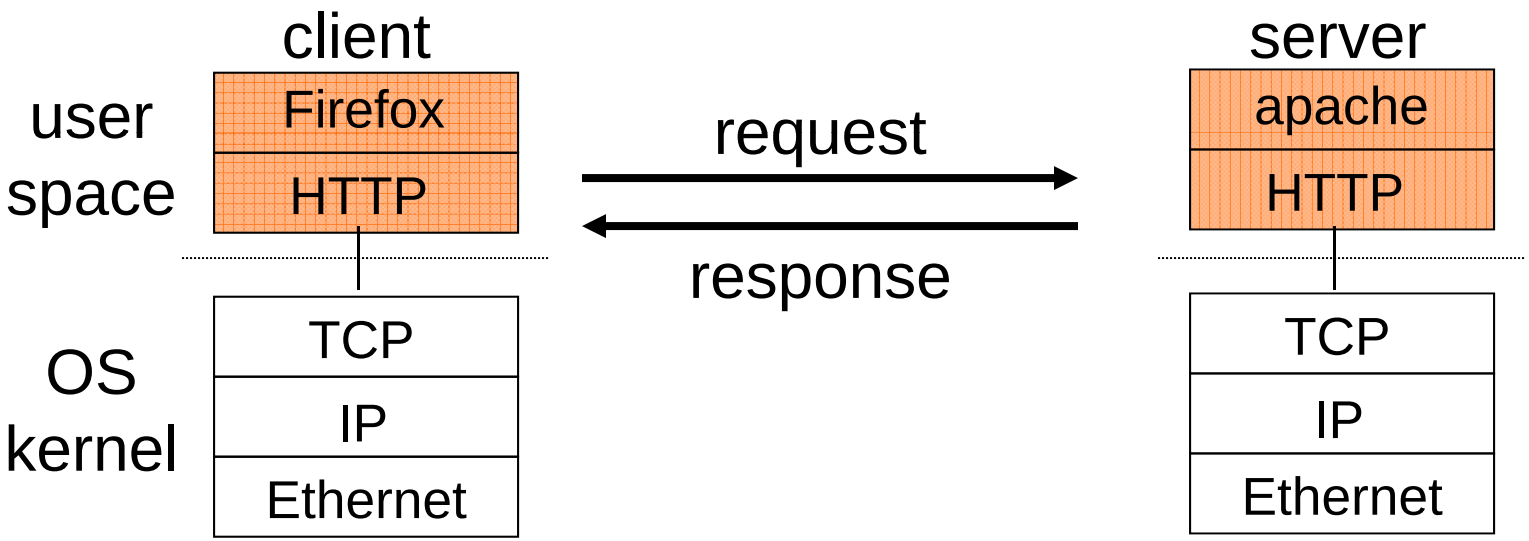
- Video conferencing
  - Unreliable video stream (congestion friendly)
- Video-on-demand (streaming media)
  - Reliable bytestream with buffered playback (congestion control)
- DNS
  - Request / reply
  - Reliable, short messages
- Web
  - Series of related request / replies
  - Reliable, variable length messages (congestion control)

- Not exactly a great match to what we have …

Web architecture

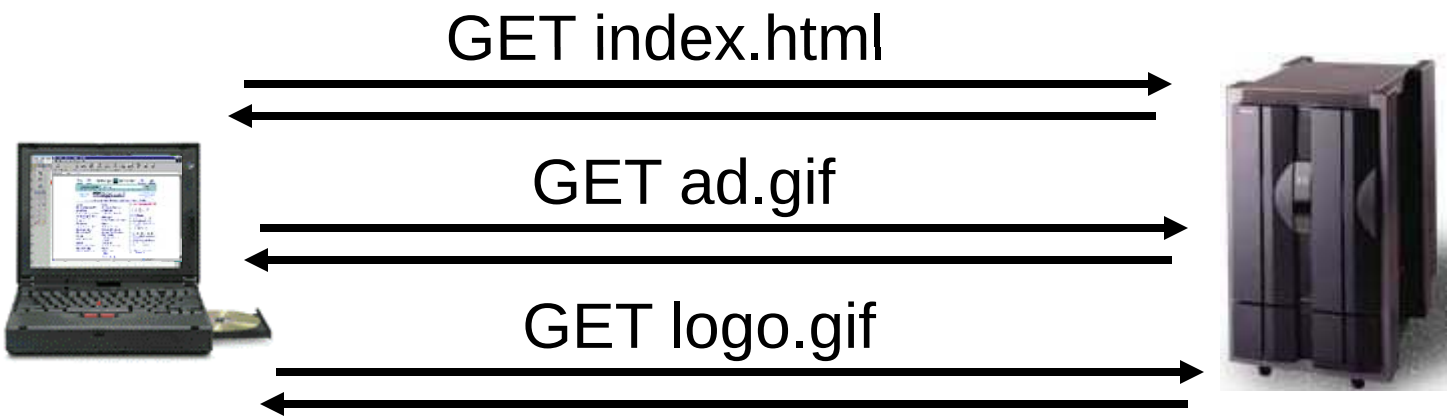


Web Protocol Stacks



- HTTP is a request/response protocol mainly used for retrieving Web resources that normally runs on TCP port 80

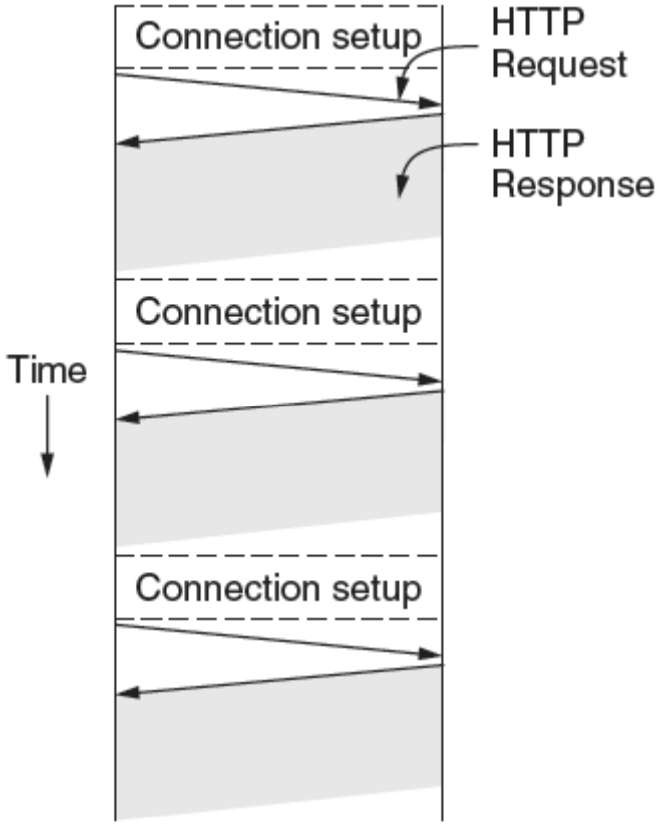
Simple HTTP 1.0



- HTTP is a tiny, text-based language
- The GET method requests an object
- There are HTTP headers, like “Content-Length:”, etc.
- Try “telnet server 80” then “GET index.html HTTP/1.0”
  - Other methods: POST, HEAD,… google for details

HTTP 1.0 Request/Responses

Time sequence diagram for a series of HTTP request/responses



HTTP 1.0 Performance

- Well, it's lower than it could be:
  - Web pages are made up of many files.
  - Most are very small (< 10k)
  - Each file is mapped to a separate TCP connection
- For each file
  - Setup/Teardown (and Time-Wait table bloat for server)
  - 2RTT "first byte" latency
  - Slow Start + AIMD Congestion Avoidance

- The goals of HTTP and TCP protocols are not aligned!

TCP Behavior for Short Connections

RTT=70ms

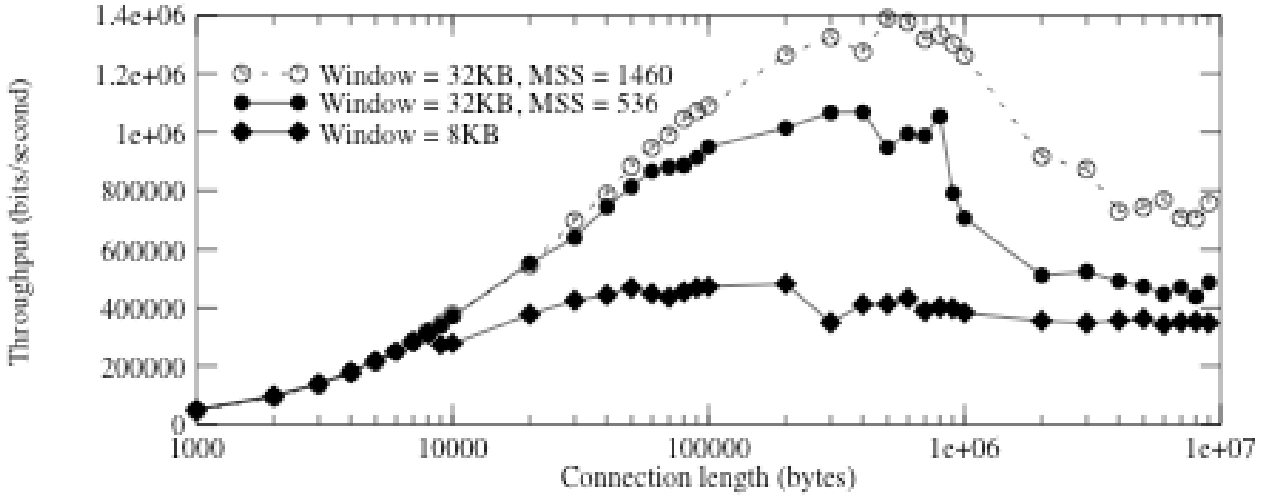


Figure 3-2: Throughput vs. connection length, RTT = 70 msec

Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.

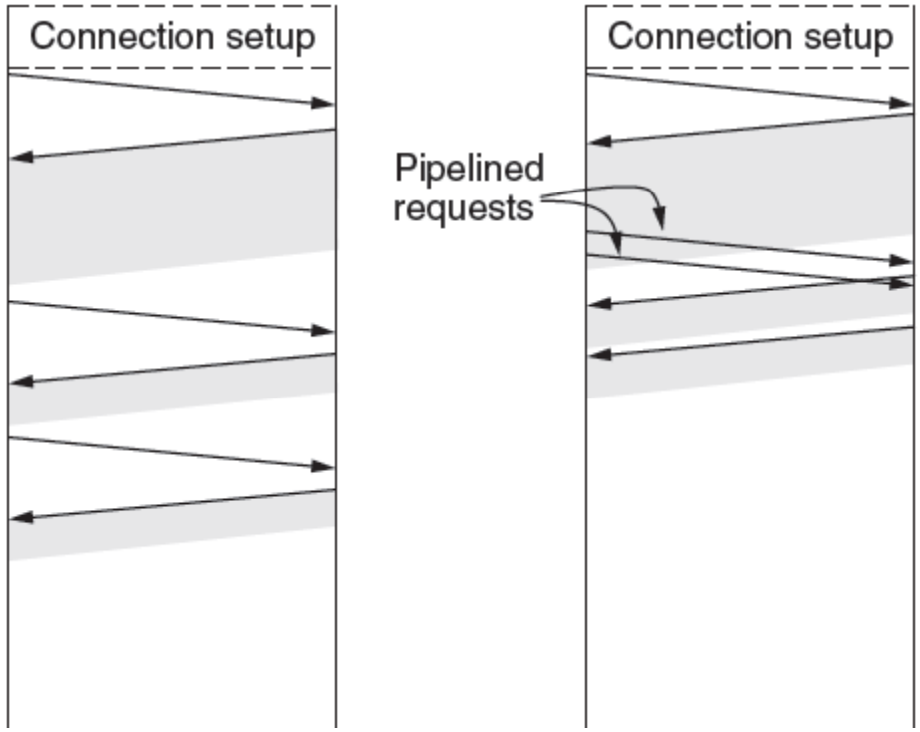
HTTP1.1: Persistent Connections

- Idea: Use one TCP connection for multiple page downloads (or HTTP methods). This is application layer multiplexing.

- Q: What are the advantages?
- Q: What are the disadvantages?

HTTP/1.1

- Only one TCP connection setup for multiple HTTP requests
- Can do HTTP request/responses serially, or pipeline them (right)
- Reduces network idle time and slow-starts



Effect of Persistent HTTP

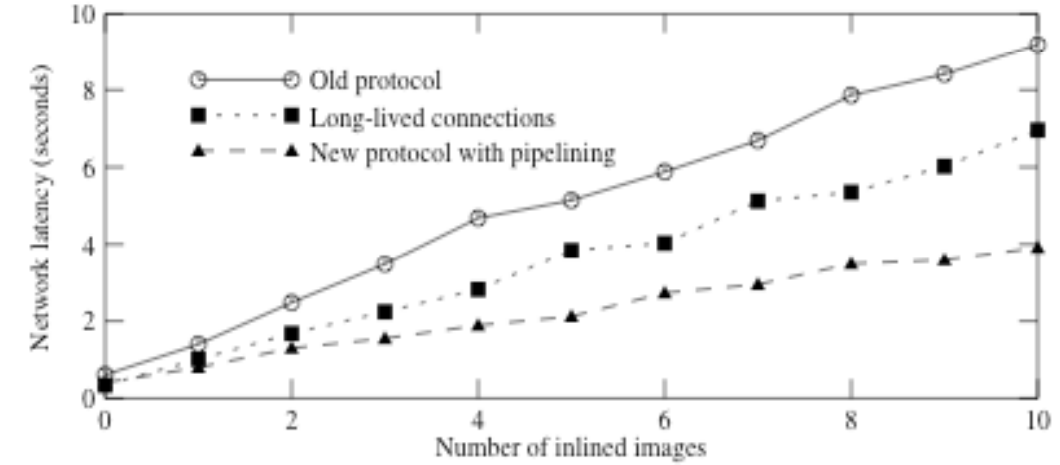


Image size=2544

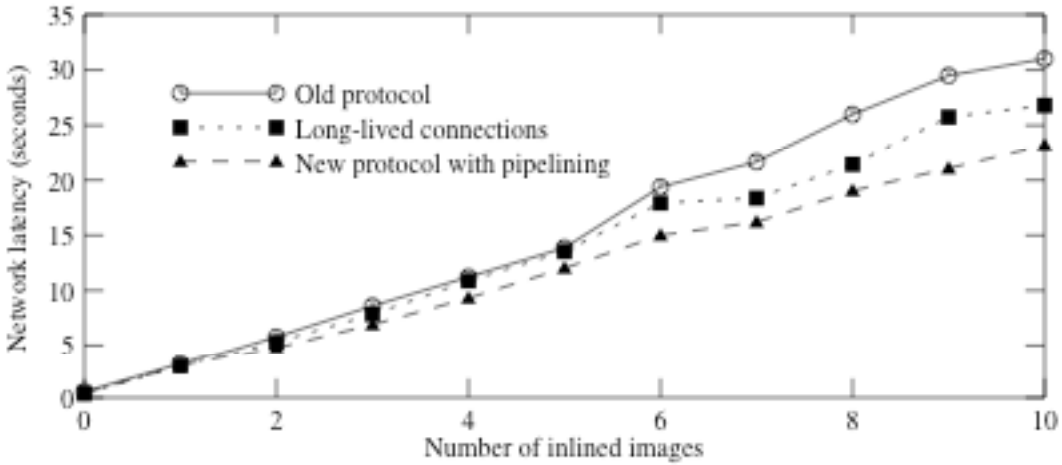


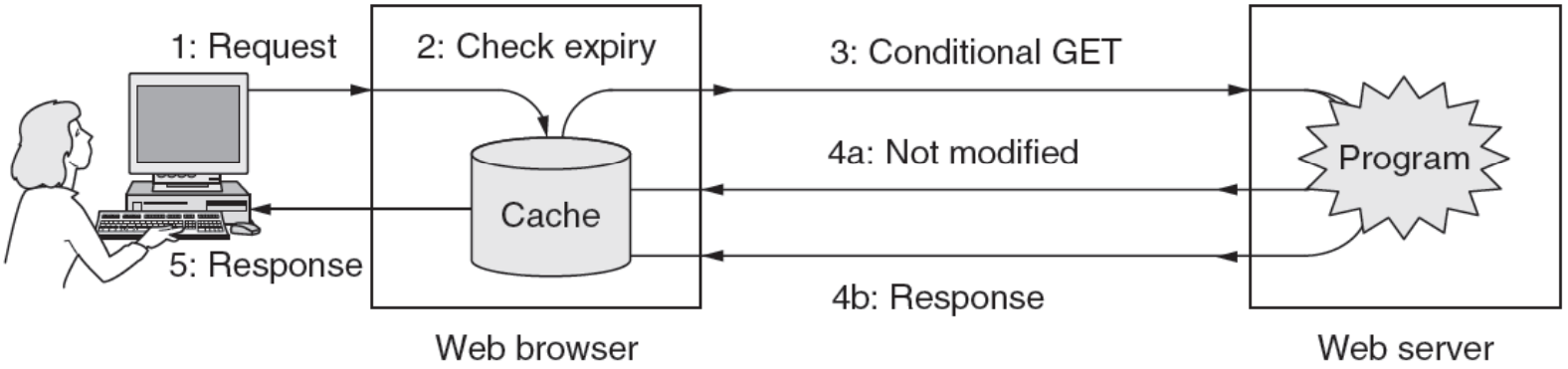
Image size=45566

Figure 6-2: Latencies for a remote server, image size = 45566 bytes

Caching

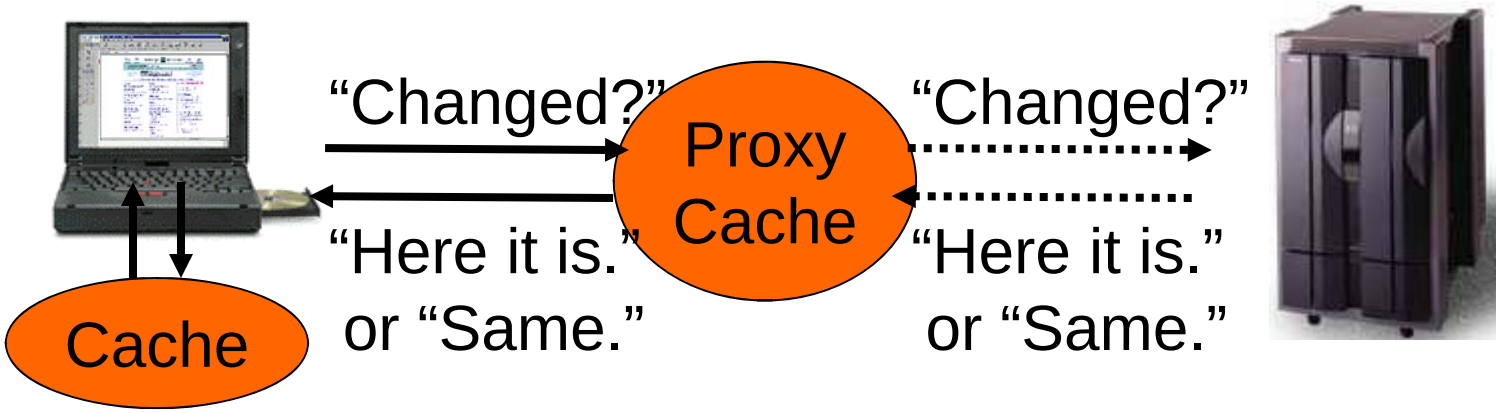
- It is faster and cheaper to get data that is closer
- Origin server may be geographically distant
- Closer can be:
  - Local browser cache (file system) (1-10ms)
  - Client-side proxy (institutional proxy) (10-50)
  - Content-distribution network (CDN -- “cloud” proxies) (50-100)

Browser Caches



- Bigger win: avoid repeated transfers of the same page
- Check local browser cache to see if we have the page
- GET with If-Modified-Since makes sure it's up-to-date

Proxy Caches



- Insert further levels of caching for greater gain
- Share proxy caches between many users (not shown)
  - If I haven't downloaded it recently, maybe you have
- Your browser has built-in support for this

Consistency and Caching Directives

- Key issue is knowing when cached data is fresh/stale
  - Otherwise many connections or the risk of staleness
- Caching directives provide hints
  - Expires: header is basically a time-to-live
  - Also indicate whether page is cacheable or not
- Browsers typically use heuristics
  - Check freshness once a "session" with GET If-Modified-Since and then assume it's fresh the rest of the time
  - Possible to have inconsistent data.

Proxy Cache Effectiveness

- Can help significantly