~13

(↓)

Computer Networks II

Application Layer

Web and Electronic Mail

Amitangshu Pal
Computer Science and Engineering
IIT Kanpur

Web and HTTP

Web and HTTP

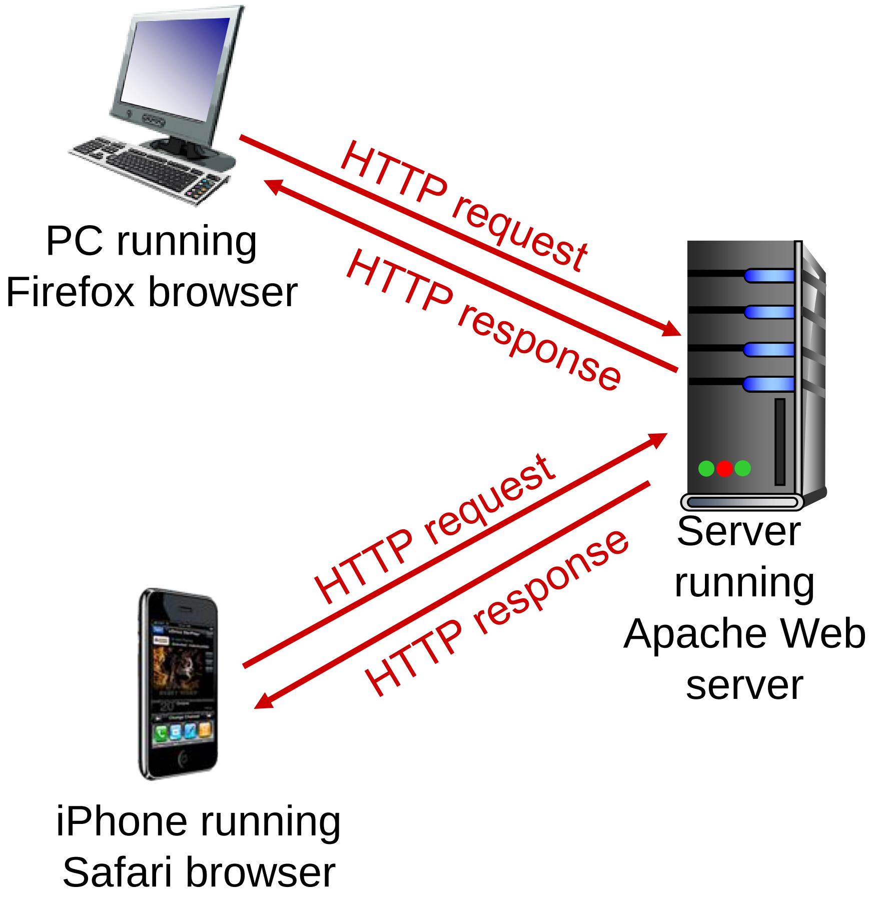
- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL, e.g.,

$$\underbrace{\text{http://}}_{\text{Protocol}}\underbrace{\text{en.Wikipedia.org}}_{\text{Server}}\underbrace{\text{/somePage/pic.gif}}_{\text{Page on server}}$$

HTTP Overview

HTTP: Hypertext transfer protocol

- Web’s application layer protocol
- Client/server model
  - Client: Browser that requests, receives, (using HTTP protocol) and “displays” Web objects
  - Server: Web server sends (using HTTP protocol) objects in response to requests



HTTP Overview

Uses TCP:

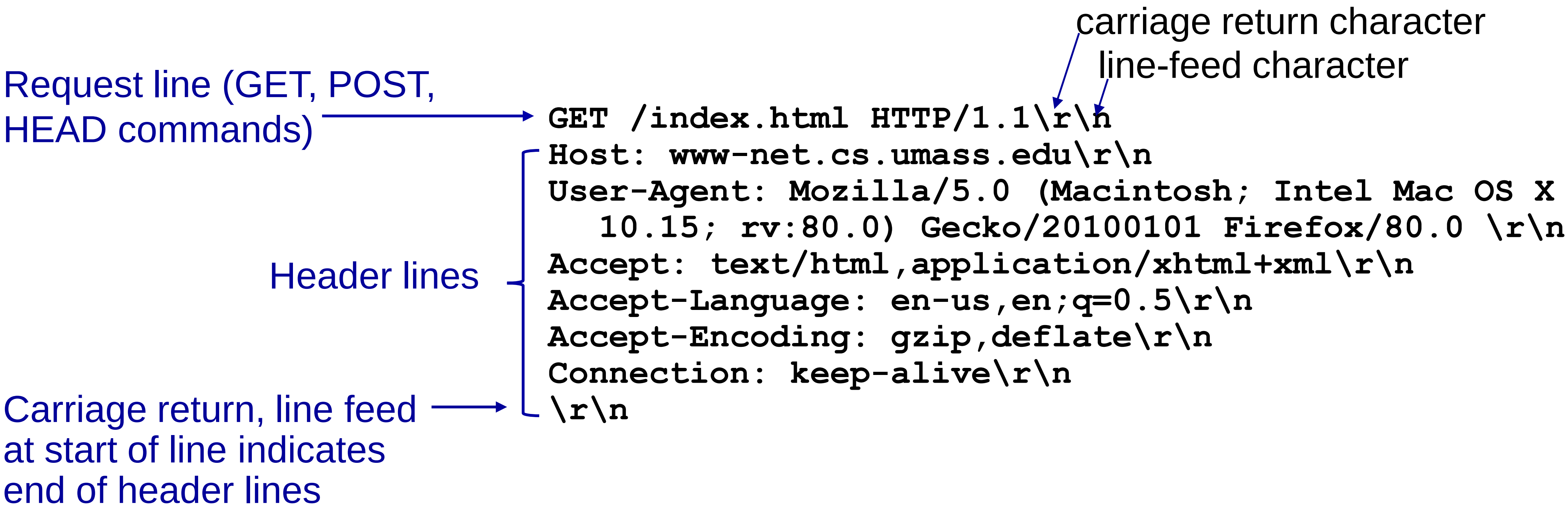
- Client resolves server to IP address (DNS)
- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is “stateless”

- Server maintains no information about past client requests

HTTP Request Message

- Two types of HTTP messages: request, response



carriage return character

line-feed character

Request line (GET, POST, HEAD commands)

Header lines

Carriage return, line feed at start of line indicates end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
   10.15; rv:80.0) Gecko/20100101 Firefox/80.0 \r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP Response Message

Status line (protocol status code status phrase)

Header lines

Data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Date: Tue, 08 Sep 2020 00:53:20 GMT
Server: Apache/2.4.6 (CentOS)
  OpenSSL/1.0.2k-fips PHP/7.4.9
  mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT
ETag: "a5b-52d015789ee9e"
Accept-Ranges: bytes
Content-Length: 2651
Content-Type: text/html; charset=UTF-8
\r\n
data data data data data ...
```

HTTP Response Status Codes

- Status code appears in 1st line in server-to-client response message

- Some sample codes:

  200 OK

  - Request succeeded, requested object later in this message

  301 Moved Permanently

  - Requested object moved, new location specified later in this message (in Location: field)

  400 Bad Request

  - Request msg not understood by server

  404 Not Found

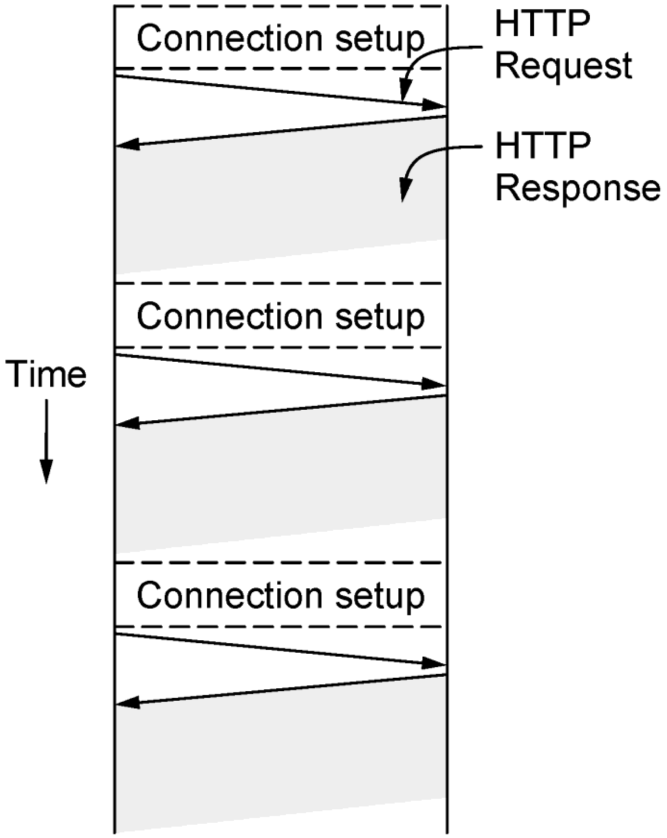
  - Requested document not found on this server

  505 HTTP Version Not Supported

Non-persistent HTTP (HTTP 1.0)

Non-persistent HTTP

- At most one object sent over TCP connection
  - Connection then closed
- Downloading multiple objects required multiple connections



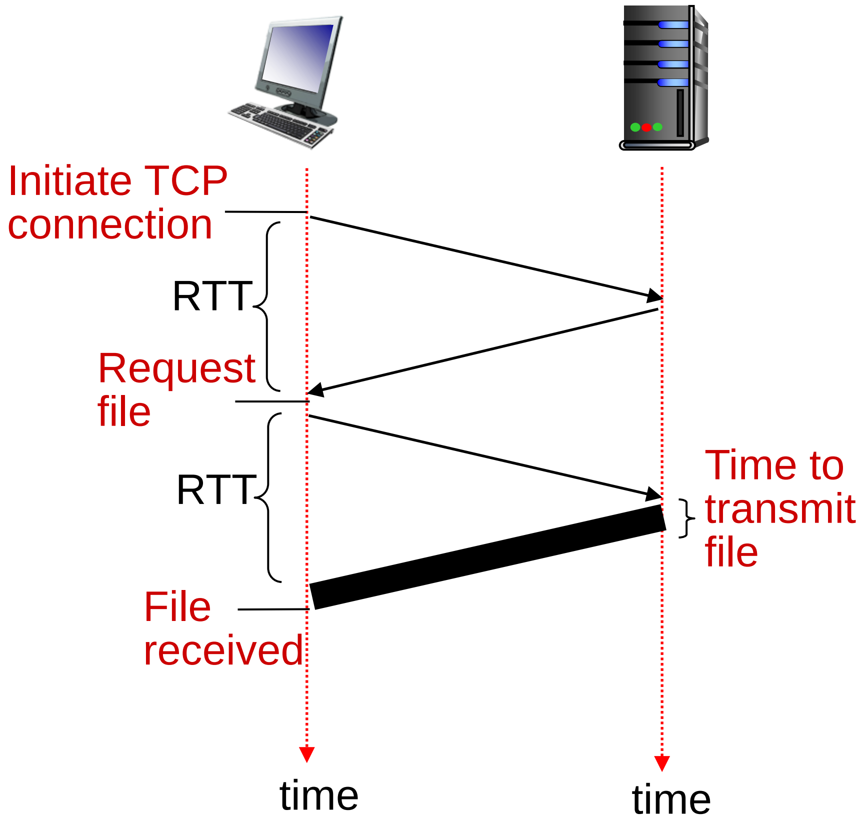
Non-persistent HTTP (HTTP 1.0): Response Time

HTTP response time:

- One RTT to initiate TCP connection
- One RTT for HTTP request and first few bytes of HTTP response to return
- File transmission time

- Non-persistent HTTP response time =

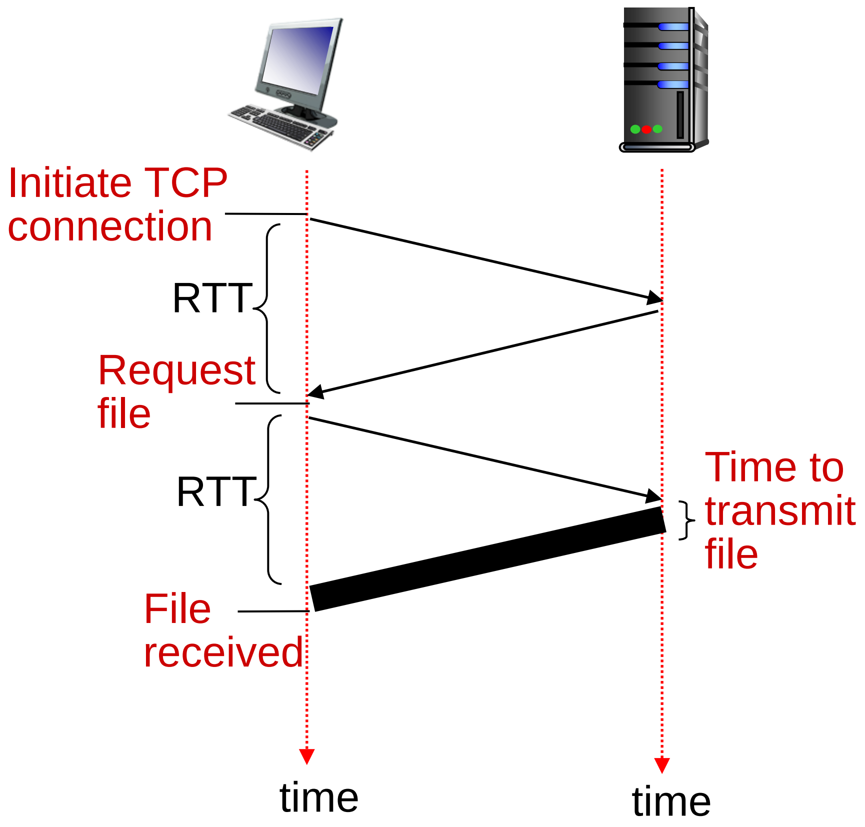
  2RTT+ file transmission time



Non-persistent HTTP (HTTP 1.0): Response Time

Non-persistent HTTP issues:

- Requires 2 RTTs per object
- OS overhead for each TCP connection
- Browsers often open parallel TCP connections to fetch referenced objects

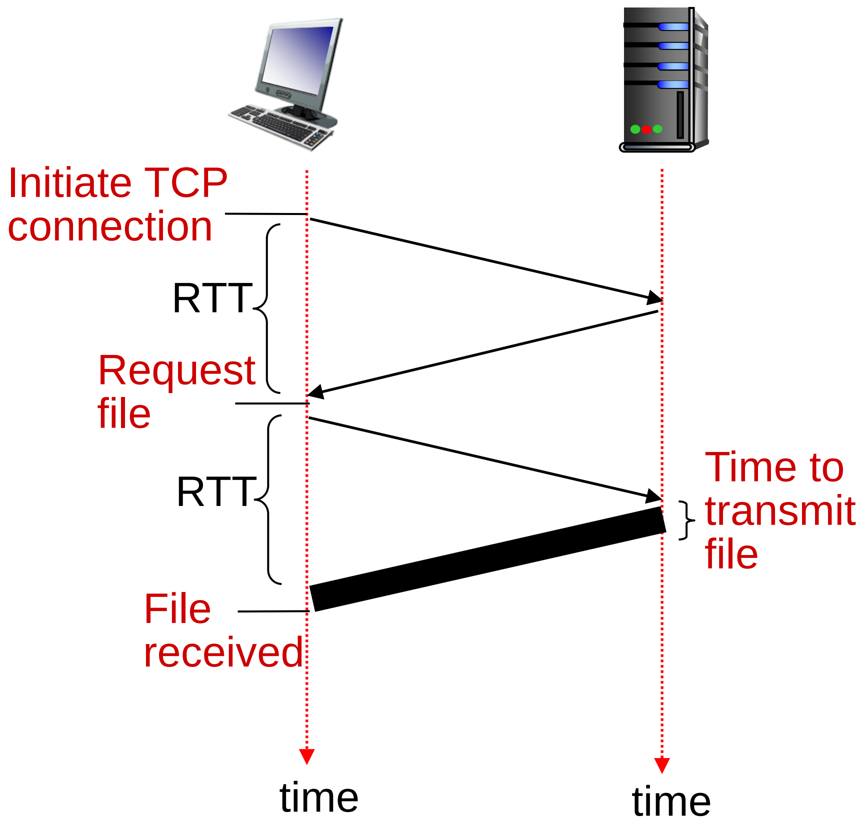


Non-persistent HTTP (HTTP 1.0): Response Time

We are downloading a webpage having a base HTML file with 5 embedded objects:

- All objects are from the same server
- The transmission time is negligible
- Objects are requested sequentially

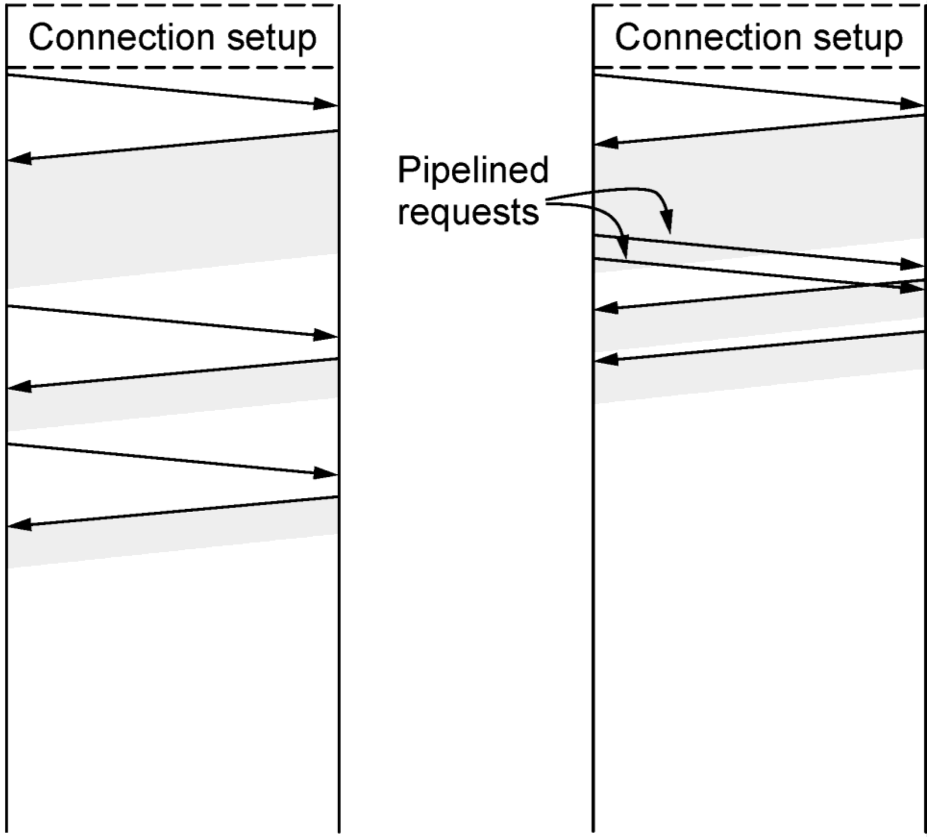
- Object requests are made in parallel



Persistent HTTP (HTTP 1.1)

Persistent HTTP

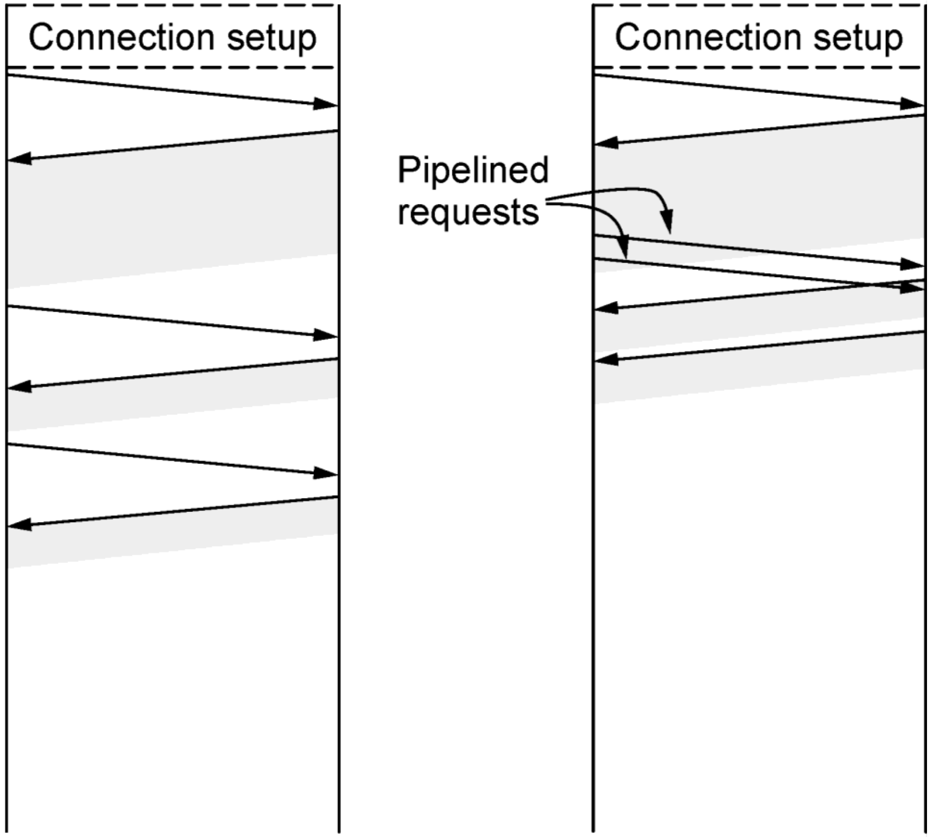
- Multiple objects can be sent over single TCP connection between client, server
- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection

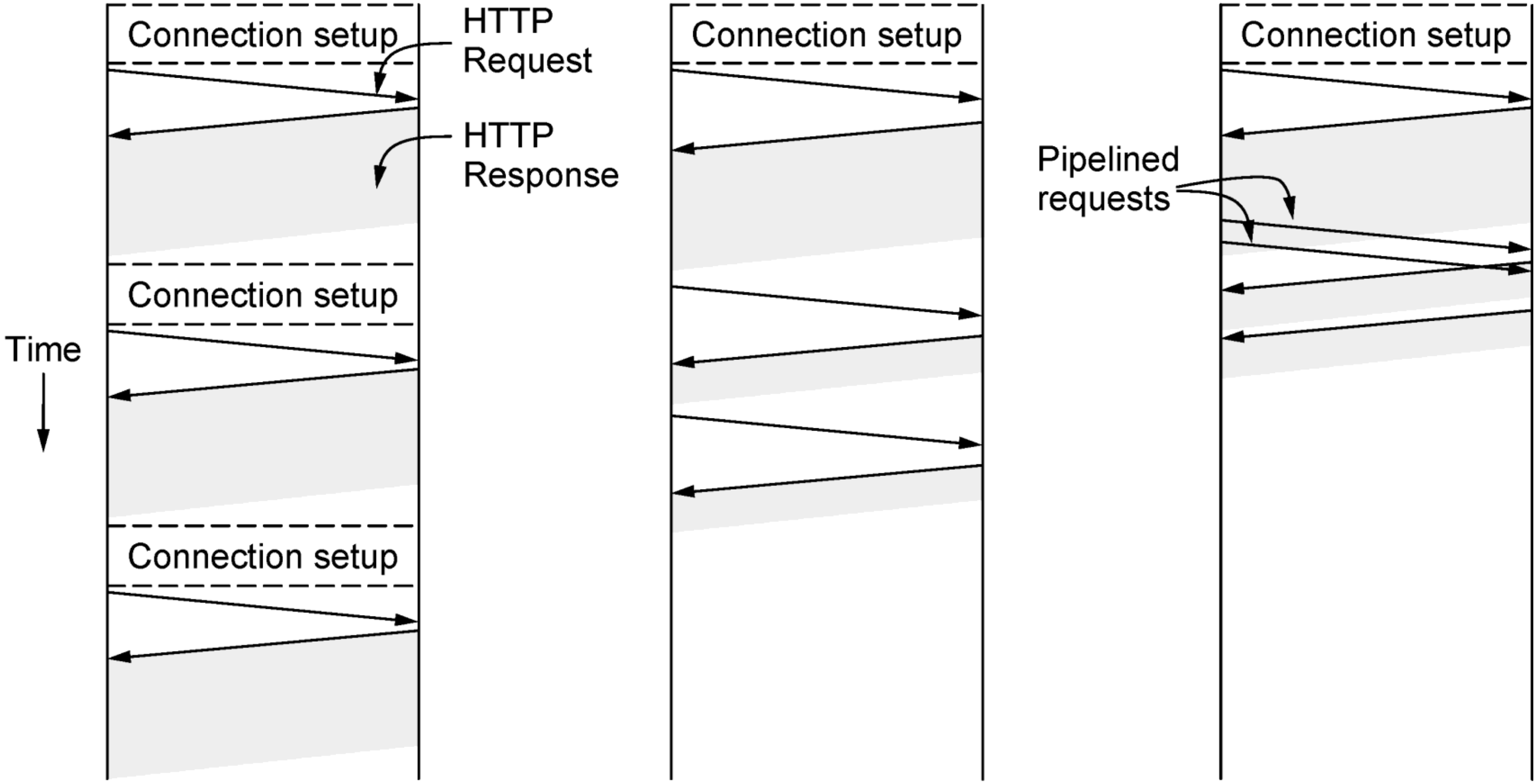


Persistent HTTP (HTTP 1.1)

Persistent HTTP

- Non-pipelined:
  - Response time of downloading a HTML file with 5 embedded objects
- Pipelined: Multiple requests can be sent together





Non-persistent HTTP

Persistent HTTP (sequential)

Persistent HTTP (pipelined)

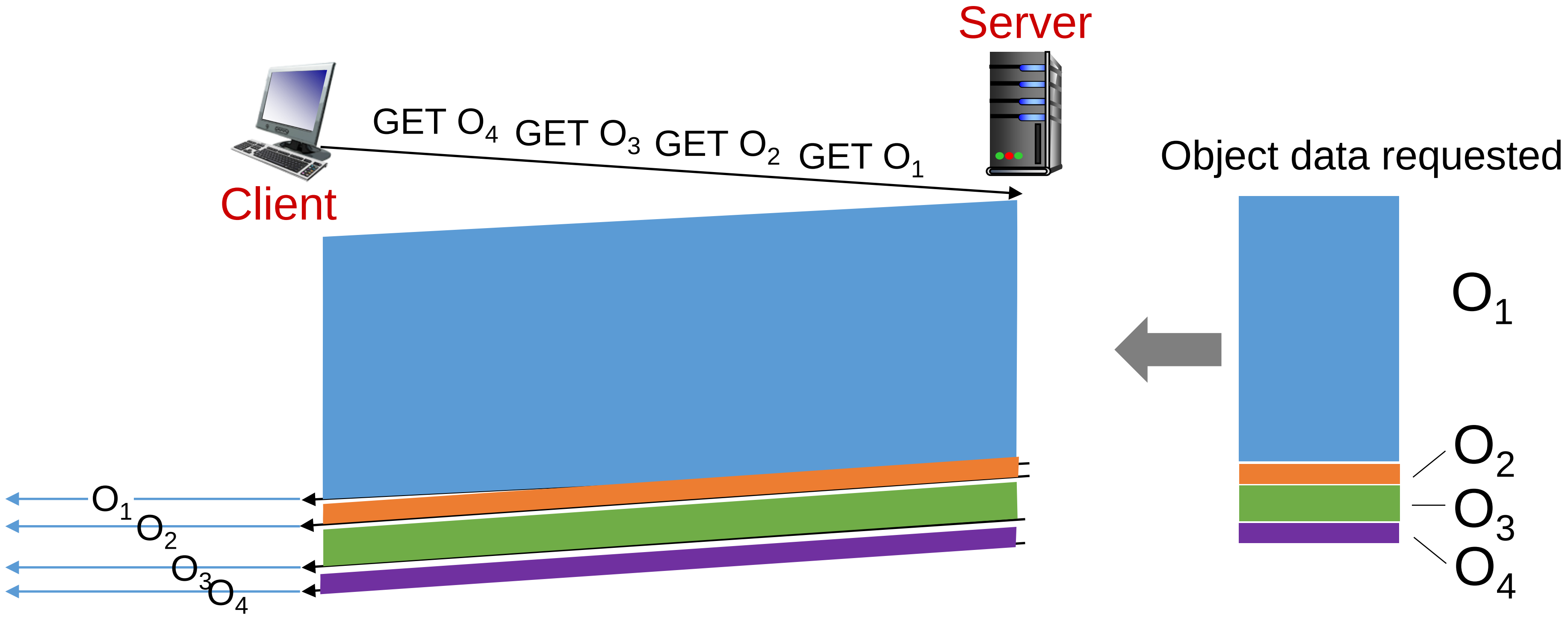
HTTP 1.1

HTTP1.1: Introduced multiple, pipelined GETs over single TCP connection

- Server responds in-order (FCFS: first-come-first-served scheduling) to GET requests
- With FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)

HTTP 1.1 (HOL Blocking)

HTTP 1.1: client requests 1 large object (e.g., video file) and 3 smaller objects



Objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

HTTP/2

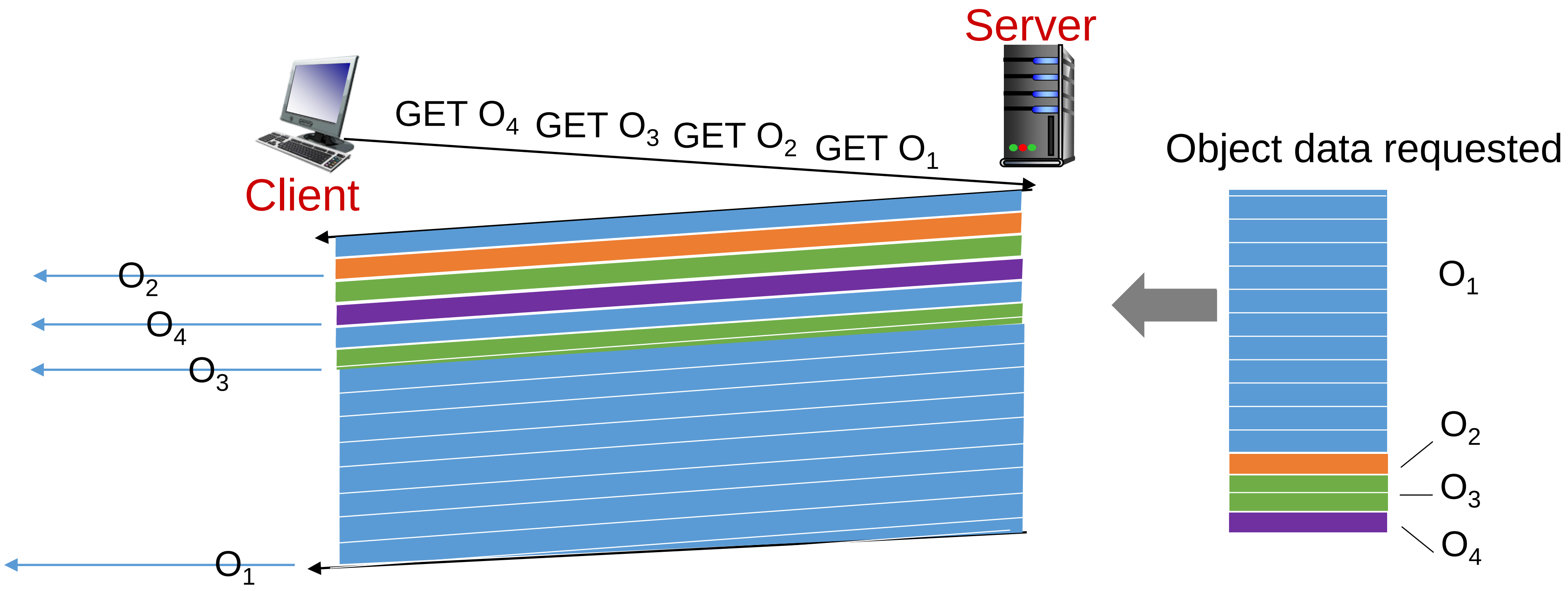
Key goal: Decreased delay in multi-object HTTP requests

HTTP/2: [RFC 7540, 2015] Increased flexibility at server in sending objects to client:

- Methods, status codes, most header fields unchanged from HTTP 1.1
- Transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- Push unrequested objects to client
- Divide objects into frames, schedule frames to mitigate HOL blocking

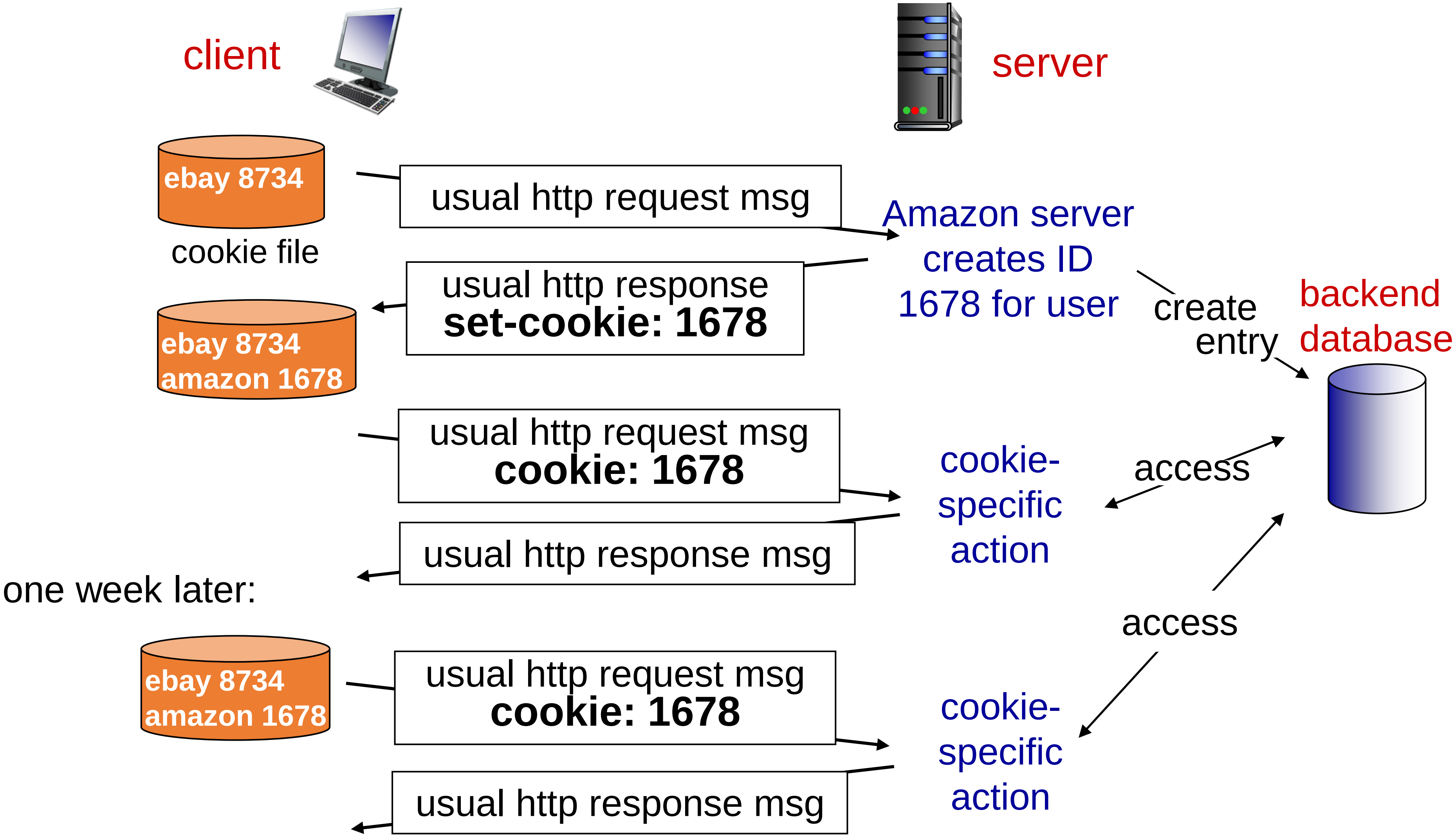
HTTP/2: Mitigating HOL Blocking

HTTP/2: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

Cookies



client

server

ebay 8734

cookie file

usual http request msg

Amazon server creates ID 1678 for user

create entry

backend database

usual http response
set-cookie: 1678

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

cookie-specific action

access

usual http response msg

one week later:

access

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

cookie-specific action

usual http response msg

User-server State: Cookies

Many Web sites use cookies

Four components:

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

Example:

- An user visits specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates:
  - Unique ID
  - Entry in backend database for ID

Cookies (continued)

What cookies can be used for:

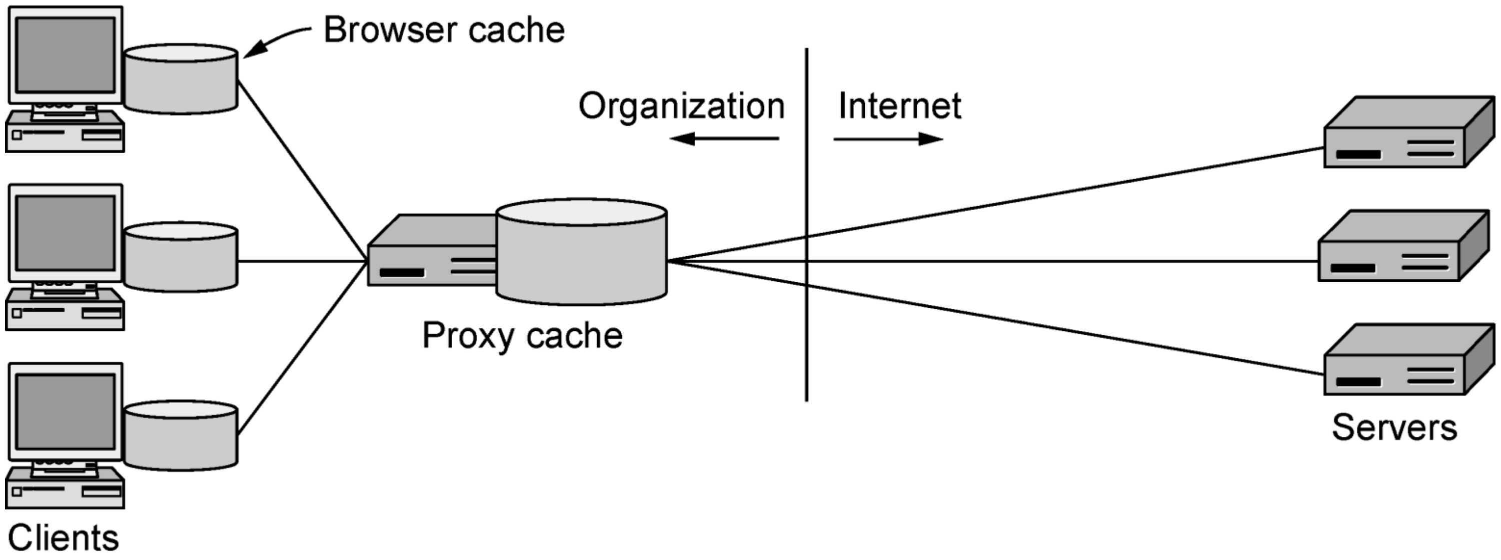
- Authorization
- Shopping carts
- Recommendations
- User session state (Web e-mail)

Web Caches

Web caching (Browser cache + Proxy cache)

User often revisit same pages

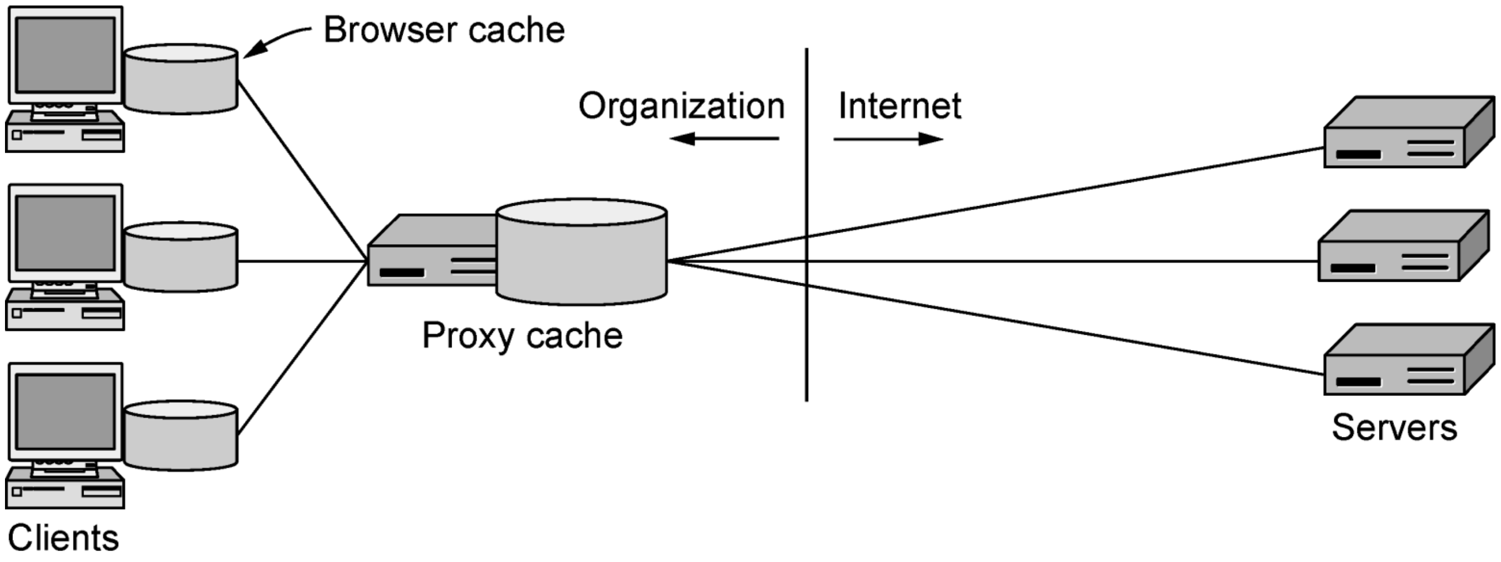
Goal: Satisfy client request without involving origin server



Web caching (Browser cache + Proxy cache)

Why Web caching?

- Reduce response time for client request
- Reduce traffic on an institution’s access link



How to verify if cached content is up-to-date?

- Server tells cache about object’s allowable caching in response header:

Cache-control: max-age = <seconds>

- Max time until the content is considered fresh

Cache-control: public

- Content is cacheable

Cache-control: private

- Content is cacheable at the browser, but not at the proxy cache

Cache-control: no-cache

- Cache but revalidate

Cache-control: no-store

- No caching

Conditional GET

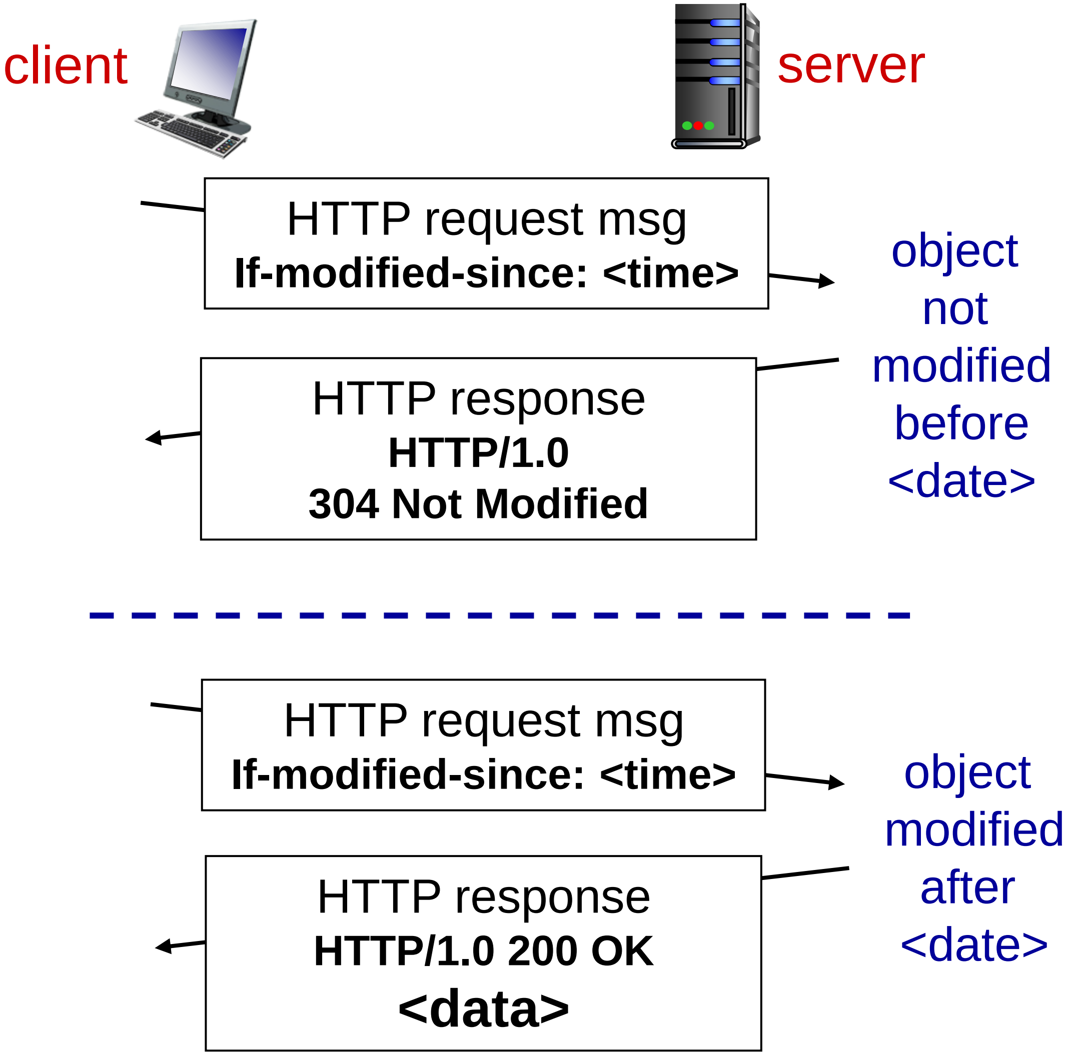
- Goal: don't send object if cache has up-to-date cached version

- Using Last-Modified + Conditional GET
- Cache: specify time of cached copy in HTTP request

  If-modified-since: <time>

- Server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



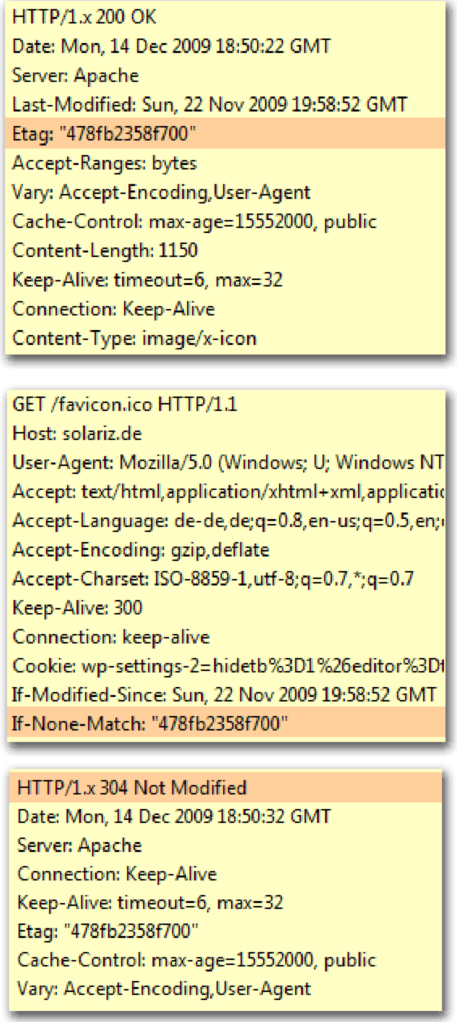
Conditional GET

- Goal: don’t send object if cache has up-to-date cached version

- Using ETag + Conditional GET
- Cache: specify Etag (hash) of cached copy in HTTP request

  If-Node-Match: <ETag>
- Server: response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



```
HTTP/1.x 200 OK
Date: Mon, 14 Dec 2009 18:50:22 GMT
Server: Apache
Last-Modified: Sun, 22 Nov 2009 19:58:52 GMT
Etag: "478fb2358f700"
Accept-Ranges: bytes
Vary: Accept-Encoding,User-Agent
Cache-Control: max-age=15552000, public
Content-Length: 1150
Keep-Alive: timeout=6, max=32
Connection: Keep-Alive
Content-Type: image/x-icon
```

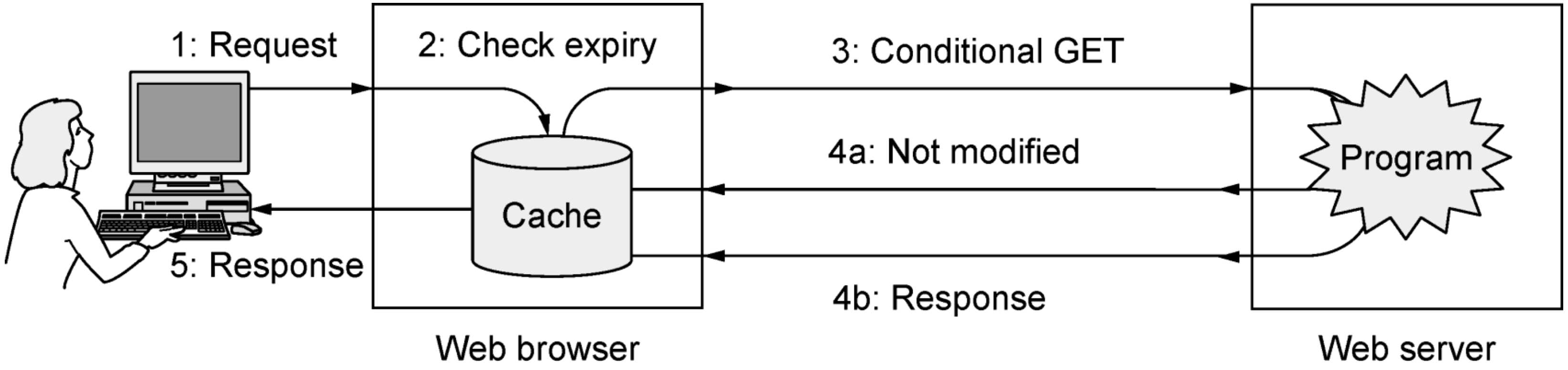
```
GET /favicon.ico HTTP/1.1
Host: solariz.de
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: wp-settings-2=hidetb%3D1%26editor%3D
If-Modified-Since: Sun, 22 Nov 2009 19:58:52 GMT
If-None-Match: "478fb2358f700"
```

```
HTTP/1.x 304 Not Modified
Date: Mon, 14 Dec 2009 18:50:32 GMT
Server: Apache
Connection: Keep-Alive
Keep-Alive: timeout=6, max=32
Etag: "478fb2358f700"
Cache-Control: max-age=15552000, public
Vary: Accept-Encoding,User-Agent
```

Src: https://commons.wikimedia.org/wiki/File:Etag_header_beispiele.png

Conditional GET

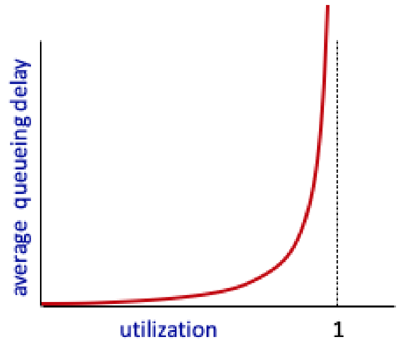
- Goal: don’t send object if cache has up-to-date cached version

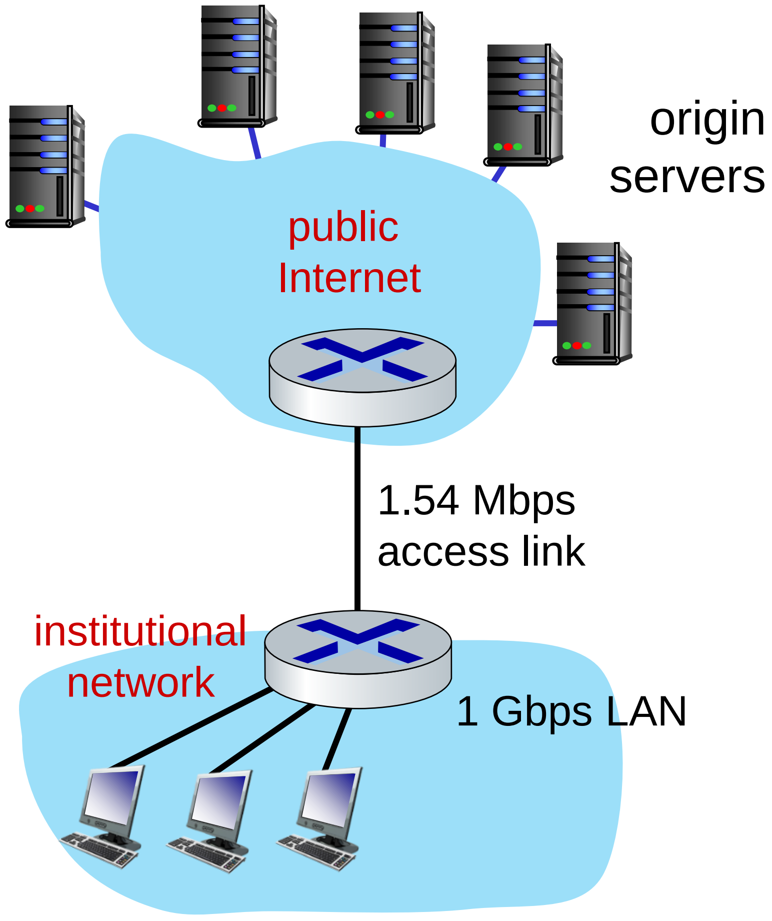


Web caching (Web Proxies)

Scenario:

- Access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - Avg data rate to browsers: 1.50 Mbps





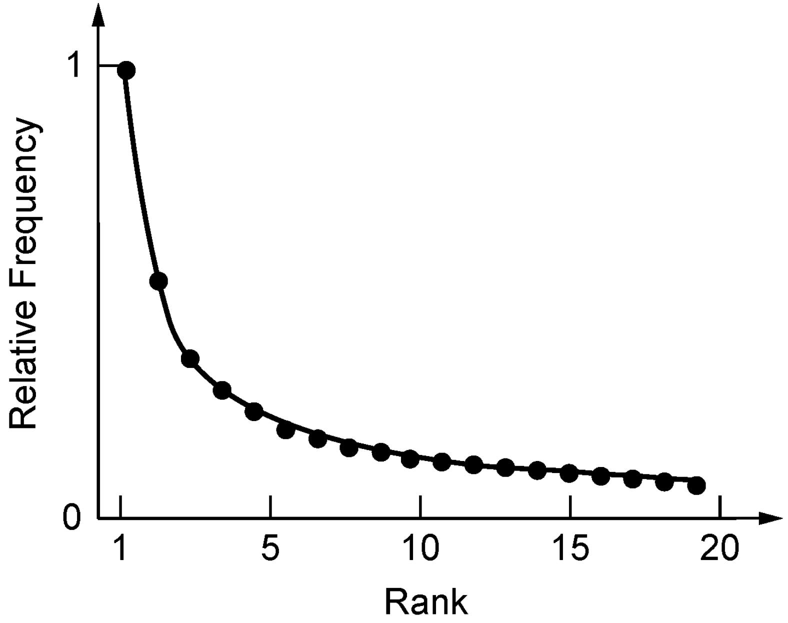
Performance:

- Access link utilization = .97
- LAN utilization: .0015
- End-end delay = RTT + access link delay + LAN delay
  = 2 sec + minutes + usecs

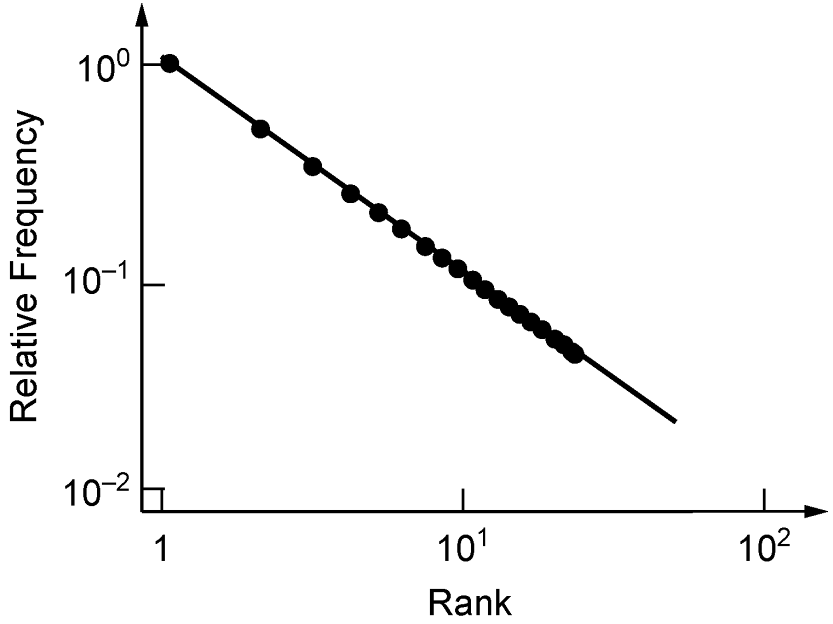
Internet Content Popularity

$$f(i,\alpha,\mathcal{M}) = \frac{\frac{1}{i^\alpha}}{\sum_{j=1}^{\mathcal{M}} \frac{1}{j^\alpha}} = \frac{\frac{1}{i^\alpha}}{\mathbf{H}_{\mathcal{M},\alpha}}$$

$$\log f(i,\alpha,\mathcal{M}) = \log\left(\frac{1}{\mathbf{H}_{\mathcal{M},\alpha}}\right) - \alpha \log i$$



Zipf distribution (linear scale)



Zipf distribution (log-log scale)

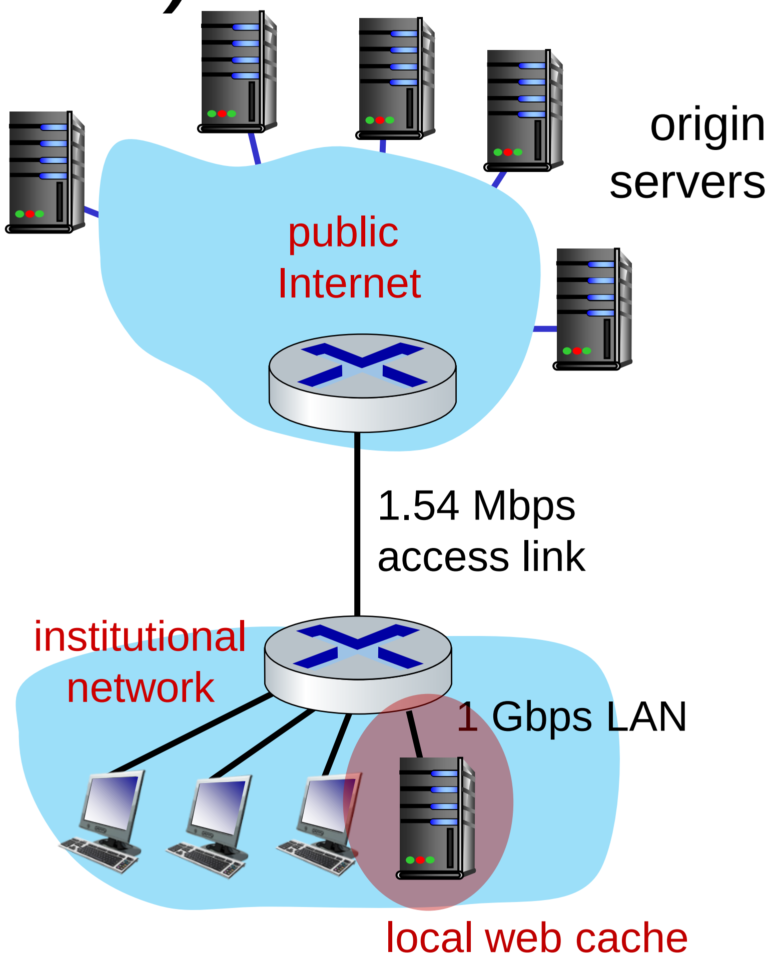
Web caching (Web Proxies)

Suppose cache hit rate is 0.4:

- 40% requests served by cache, with low (msec) delay
- 60% requests satisfied at origin
  - Rate to browsers over access link

    = 0.6 * 1.50 Mbps = .9 Mbps
  - Access link utilization = 0.9/1.54 = .58 means low (msec) queueing delay at access link
- Average end-end delay:

  = 0.6 * (delay from origin servers) + 0.4 * (delay when satisfied at cache) = 0.6 (2 secs) + 0.4 (~msecs) = ~ 1.2 secs



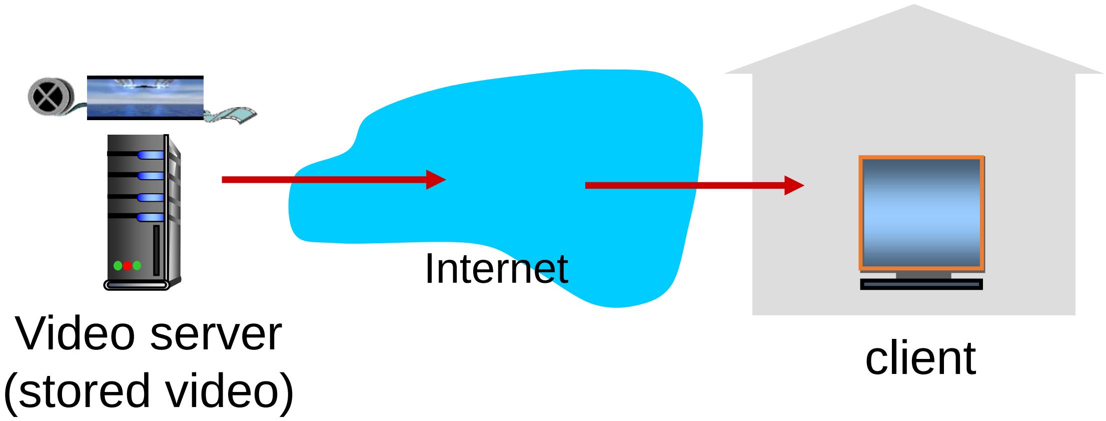
Content Distribution Networks (CDNs)

Video Streaming and CDNs

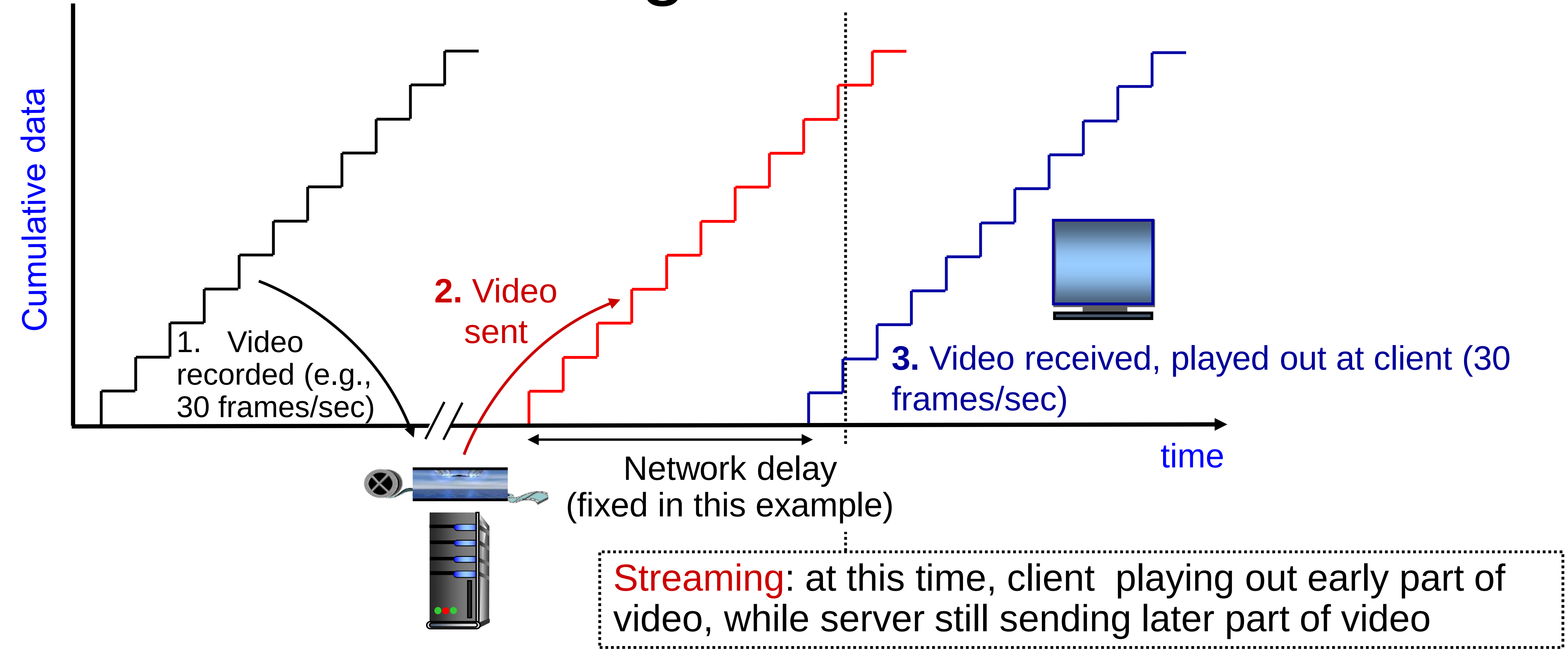
- Stream video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube, Amazon Prime: 80% of residential ISP traffic (2020)

- Challenges:
  - Scale - how to reach ~1B users?
  - Heterogeneity
    - Different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)

Streaming Stored Video



Streaming Stored Video



Cumulative data

1. Video recorded (e.g., 30 frames/sec)

2. Video sent

3. Video received, played out at client (30 frames/sec)

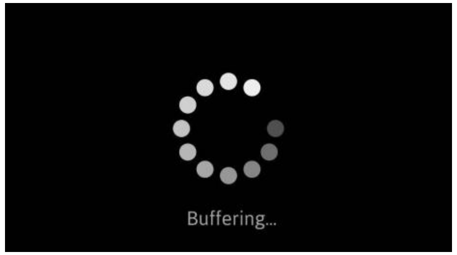
Network delay (fixed in this example)

time

Streaming: at this time, client playing out early part of video, while server still sending later part of video

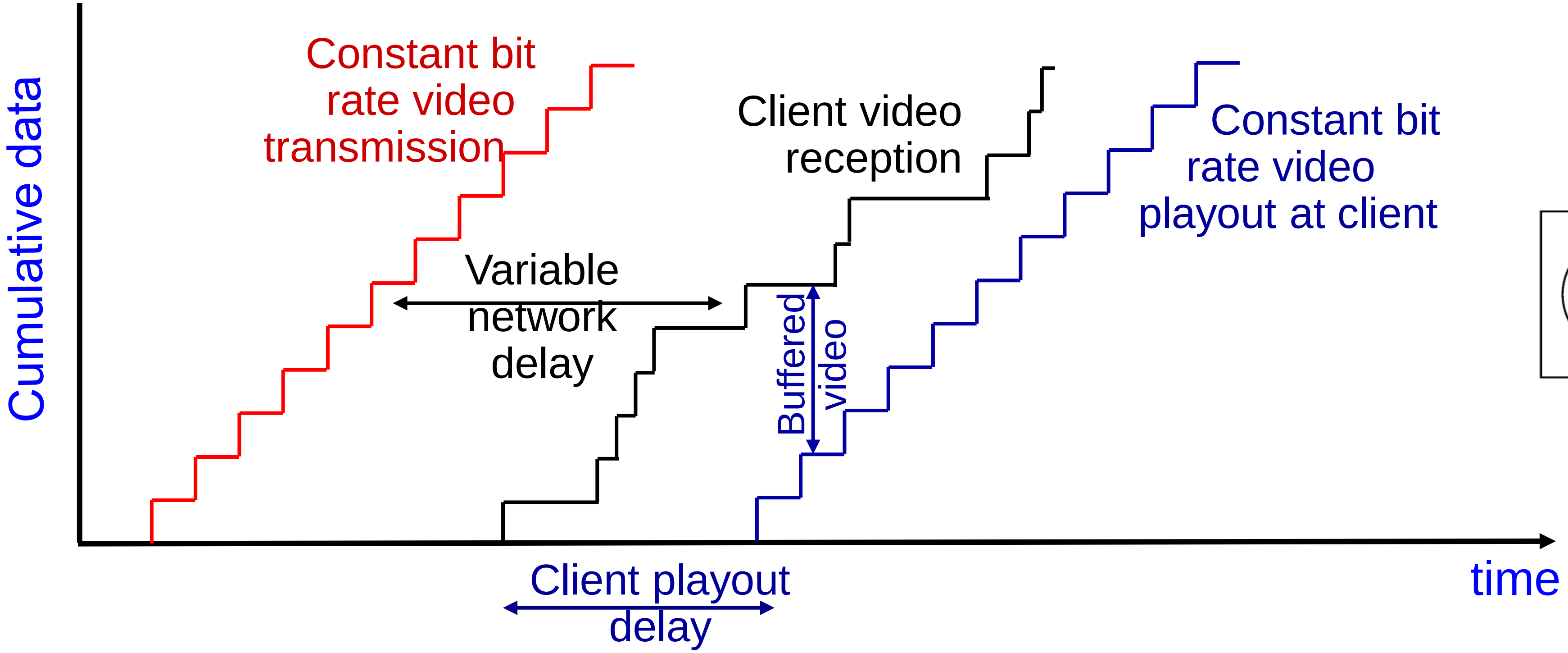
Streaming Stored Video: Challenges

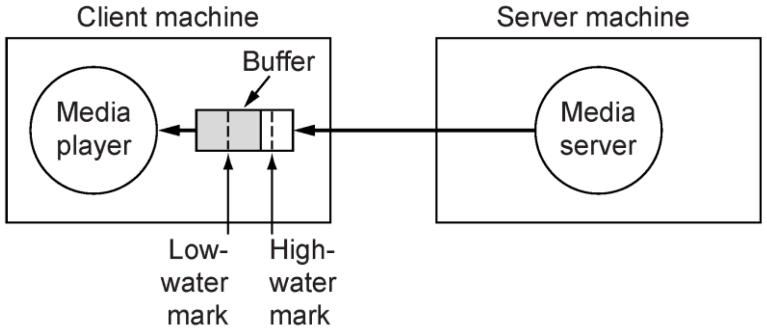
- Continuous playout constraint: During client video playout, playout timing must match original timing
  - But network delays are variable (jitter), so will need client-side buffer to match continuous playout constraint



- Other challenges:
  - Client interactivity: pause, fast-forward, rewind, jump through video
  - Video packets may be lost, retransmitted

Streaming Stored Video: Playout Buffering





- Client-side buffering and playout delay: compensate for network-added delay, delay jitter

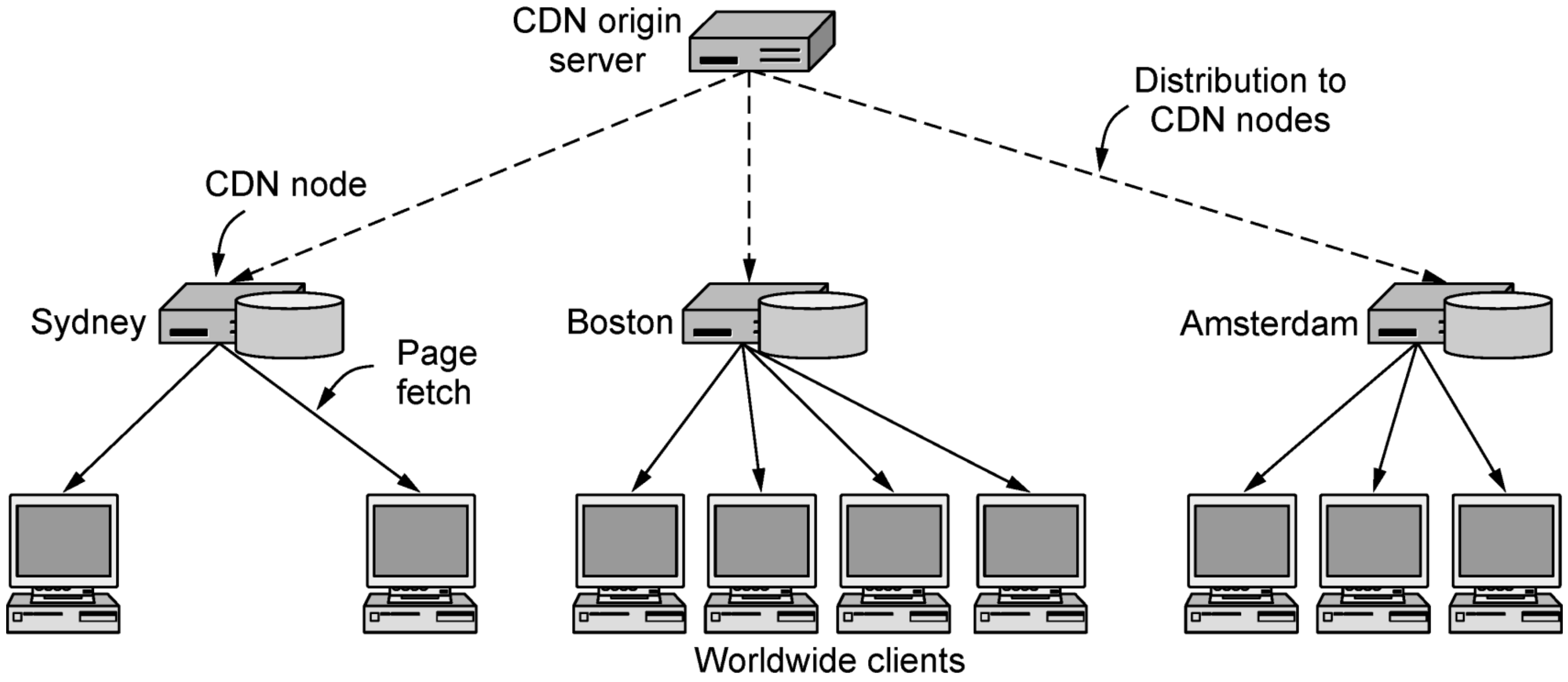
Content Distribution Networks (CDNs)

Challenge: How to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

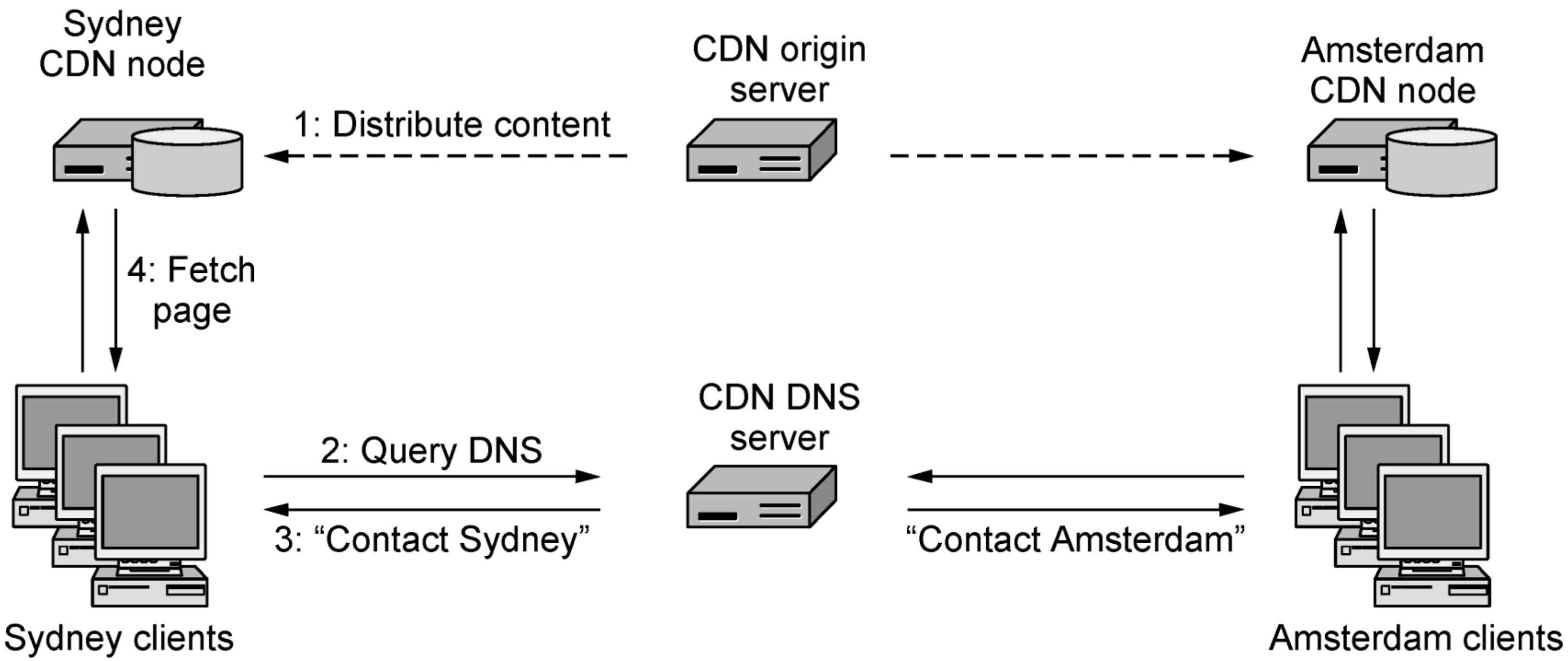
- Solution: Store/serve multiple copies of videos at multiple geographically distributed sites (CDN)

- Push CDN servers deep into many access networks
  - Close to users
  - Akamai: 240,000 servers deployed in > 120 countries (2015)

Content Distribution Networks (CDNs)



Content Distribution Networks: DNS Redirection



http://www.Netflix.com/page.html

Content Distribution Networks (CDNs)

Win-win solution for all parties:

- Reduces server load → Content providers are 🙂
- Reduces network load → ISPs are 🙂
- Reduces content download time → Users/customers are 🙂

Electronic Mail

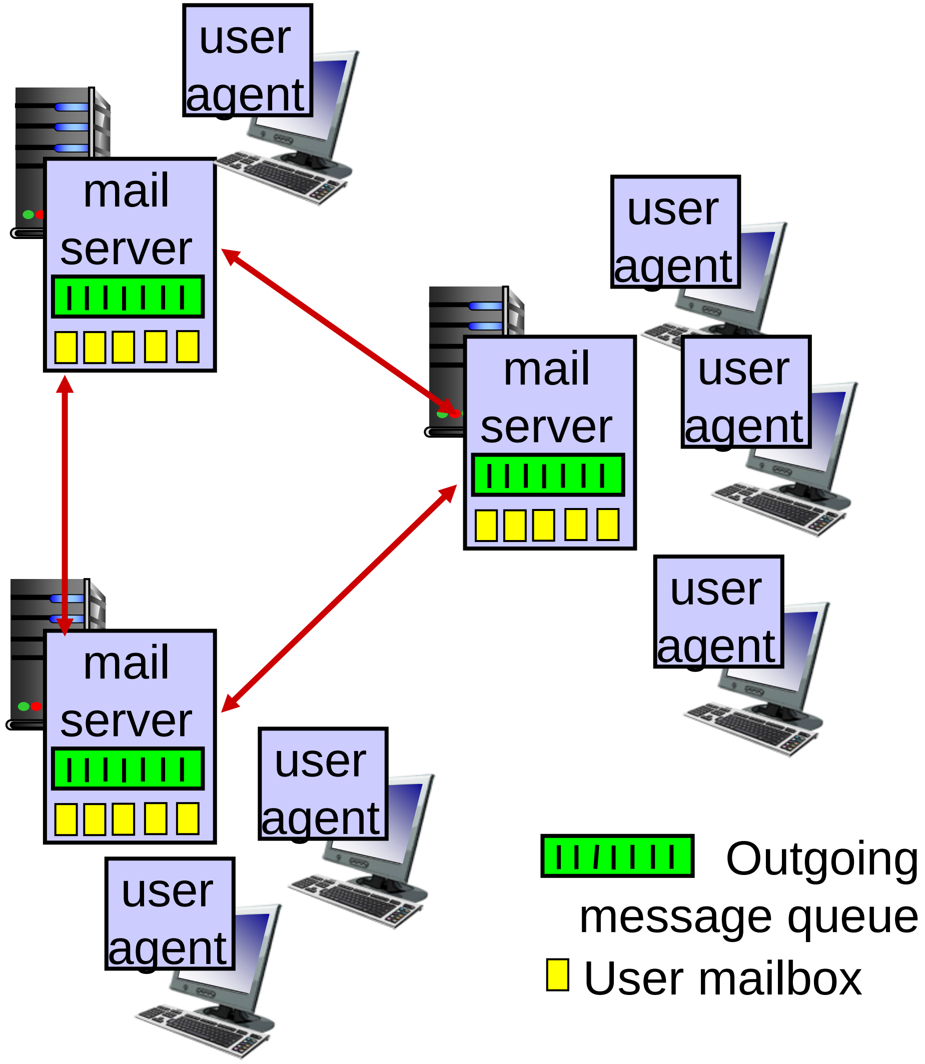
Electronic Mail

User Agent

- a.k.a. “mail reader”
- Composing, editing, reading mail messages

Mail servers:

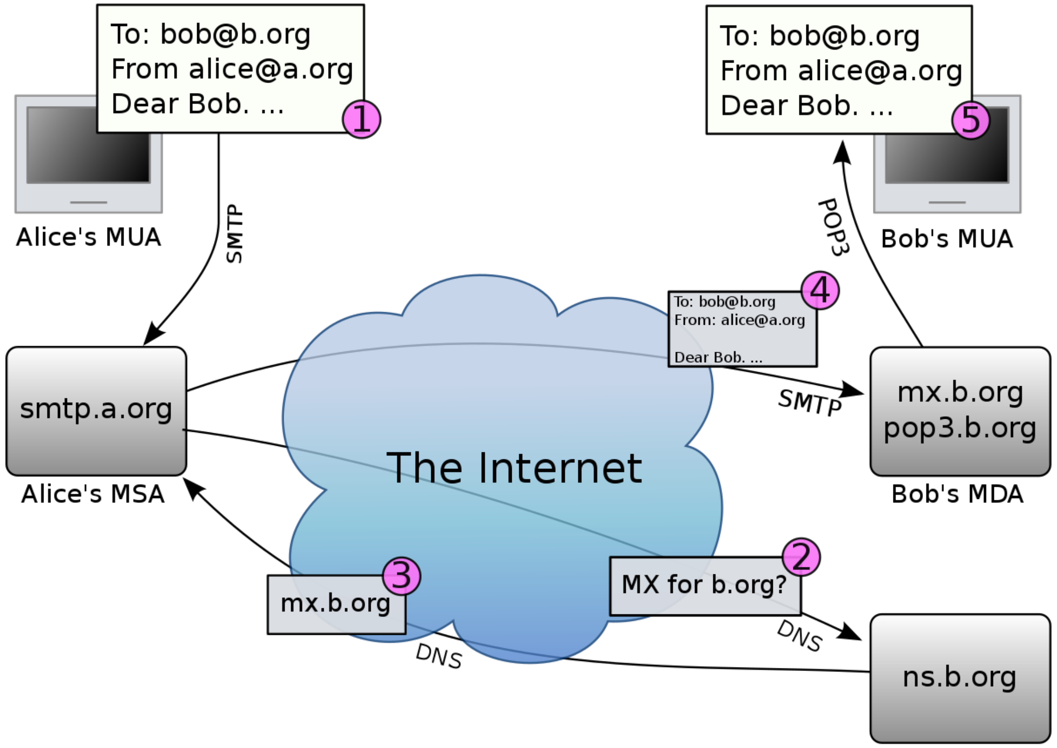
- Mailbox contains incoming messages for user
- Message queue of outgoing (to be sent) mail messages



Electronic Mail

- Mail transfer protocol: Delivery/storage to receiver’s server → Push operation
  - SMTP: Simple Mail Transfer Protocol
  - User agent to mail server
  - In between mail servers

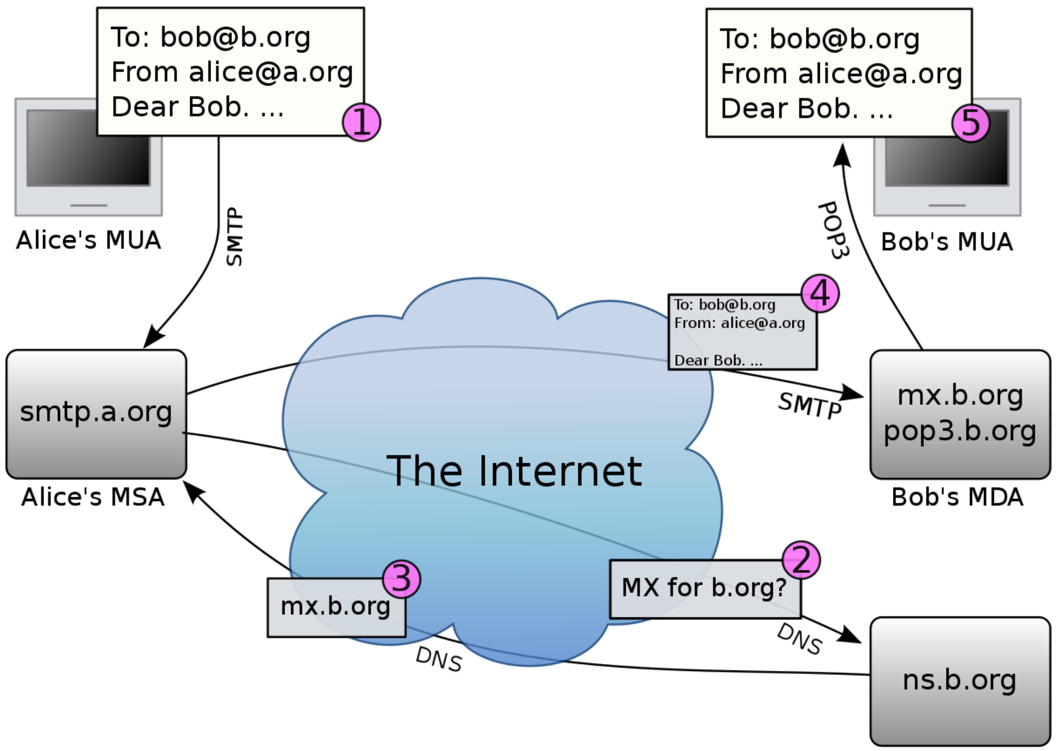
- SMTP protocol between mail servers to send email messages
  - Client: sending mail server
  - “Server”: receiving mail server



Src: https://en.wikipedia.org/wiki/Open_mail_relay#/media/File:Email.svg

Electronic Mail

- Mail access protocol: retrieval from server → Pull operation
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

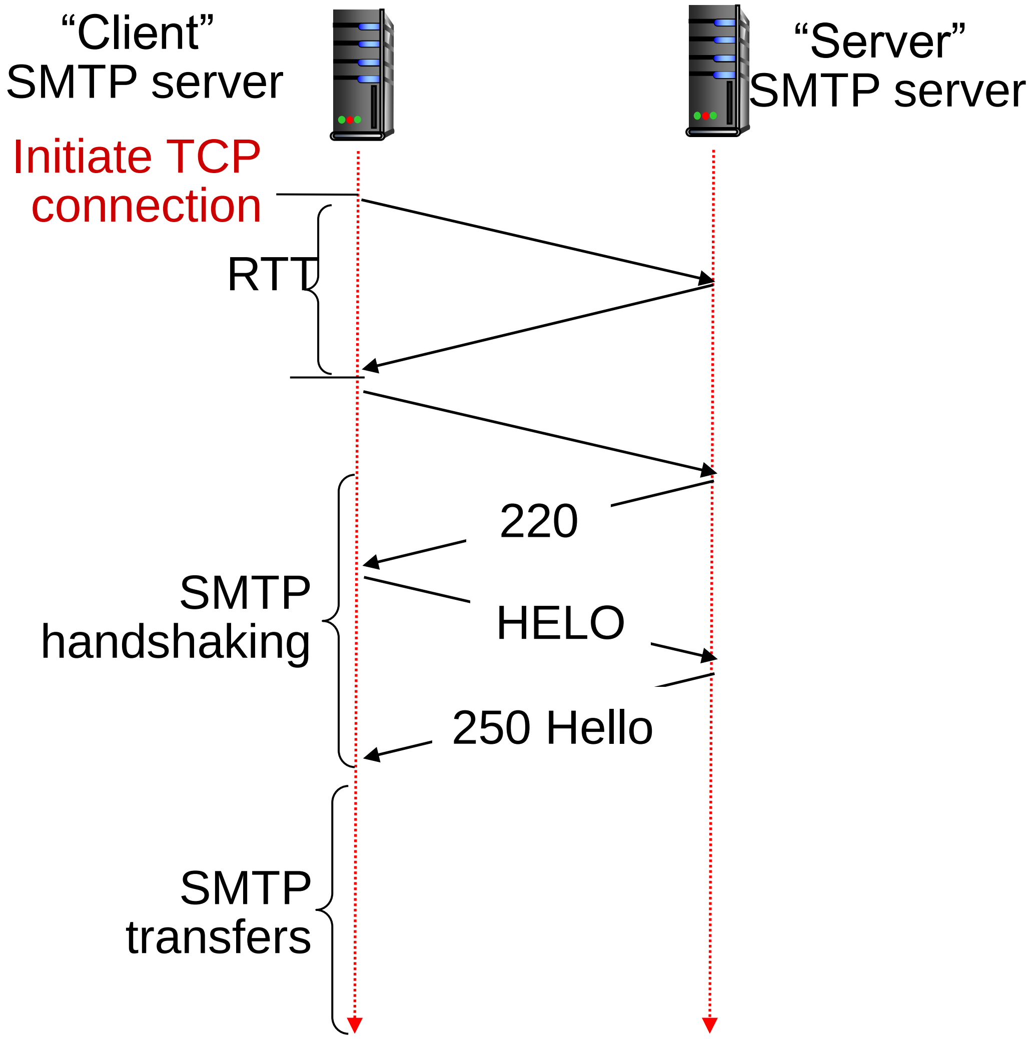


Src: https://en.wikipedia.org/wiki/Open_mail_relay#/media/File:Email.svg

SMTP

- Uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25
  - Direct transfer: sending server (acting like client) to receiving server

- Three phases of transfer
  - SMTP handshaking (greeting)
  - SMTP transfer of messages
  - SMTP closure



Summary

❑ Web and HTTP:

- Persistent and non-persistent HTTP
- Web cache

❑ Electronic mail:

- SMTP, POP, IMAP