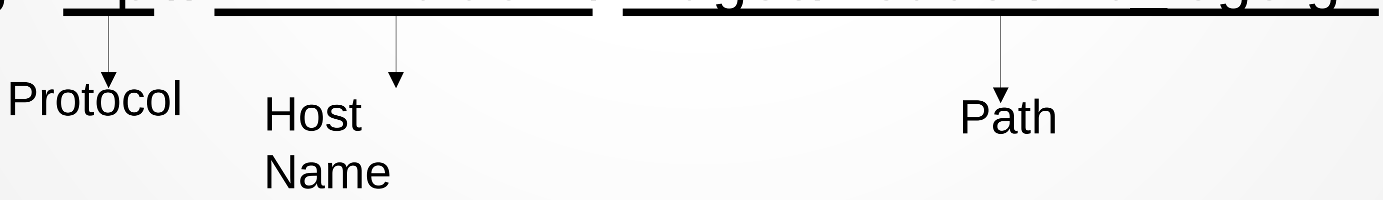
Web and HTTP

Manas Ranjan Lenka
School of Computer Engineering,
KIIT University

Web

- Web pages are viewed by a program called a browser
  - E.g. Internet Explorer, Google Chrome, Mozilla Firefox
- Web page consists of base HTML file which includes several referenced objects
  - Object can be other HTML files, JPEG images, Java applets, audio files,......
  - Text/Image that links to another page is called a hyperlink (often highlighted by some means)
- Each object is addressable by a URL (Uniform Resource Locator)
  - E.g. http://www.iitb.ac.in/images/header/iitb_logo.gif



Hyper Text Transfer Protocol (HTTP)

- Based on client-server model
  - Client (browser) requests web objects
  - Server responds with status code and requested object (if present)
- Operates over TCP, server port 80
- Two Versions:
  - HTTP 1.0 (RFC 1945)
  - HTTP 1.1: (RFC 2068)
- Stateless protocol: no user information stored across requests

HTTP Non-persistent Connection

- Used by HTTP/1.0
- At most one object is sent over a TCP connection
- Rather inefficient in terms of operating system overhead (especially at server) and response time
  - Response Time: Time when a request was made and the object fully received
  - Takes 2RTT +TX-time per object

Example

- Download a html webpage with 5 embedded objects

- What is the overall response time to display the webpage fully?

  - Assume object fits within one packet
  - Assume objects requests are made sequentially
    - Total Time is 2RTT + 5*2RTT = 12 RTT

– What if the object requests are made parallelely?

  - Total time is ~ 2RTT + 2RTT = 4RTT

HTTP Persistent Connections

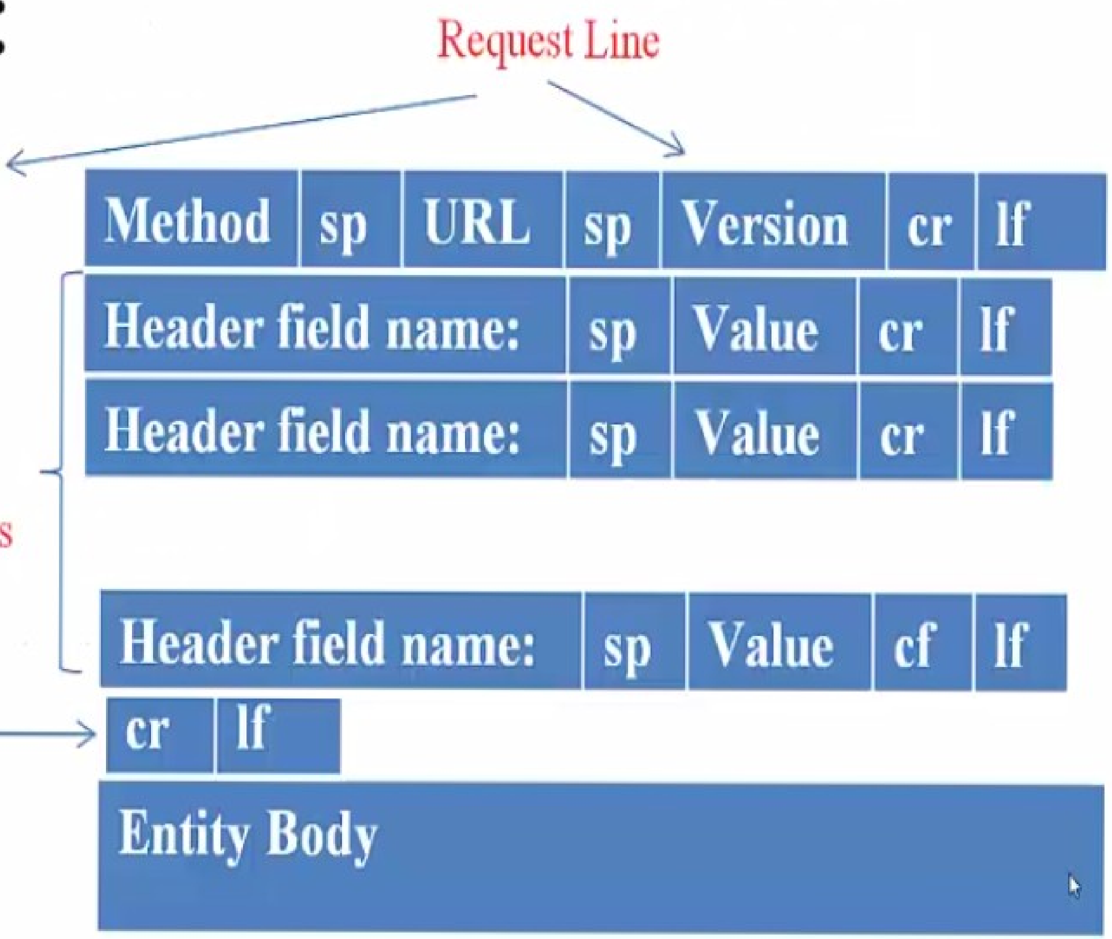
- Used by HTTP 1.1
- Server connection left open for subsequent requests
  - Helps reduce TCP related overhead (buffers, state etc) at server
- Two modes of operation:
- Non-pipelined: new request sent only after previous request completes
  - Example: html page with 5 embed object
    - Total time: 2RTT+ 5RTT = 7RTT
- Pipelined: Multiple requests can be sent at once; default mode of operation
  - Minimum total time: 2RTT+RTT = 3RTT

HTTP Message Format

- Two types of messages: Request and Response
- Request Message:

GET /~chebrolu/ HTTP/1.1
Host: www.cse.iitb.ac.in
User-agent: Mozilla/5.0
Connection: close
Accept-language: fr
(blank line)

Header Lines



Request Line

Method sp URL sp Version cr lf
Header field name: sp Value cr lf
Header field name: sp Value cr lf
Header field name: sp Value cf lf
cr lf
Entity Body

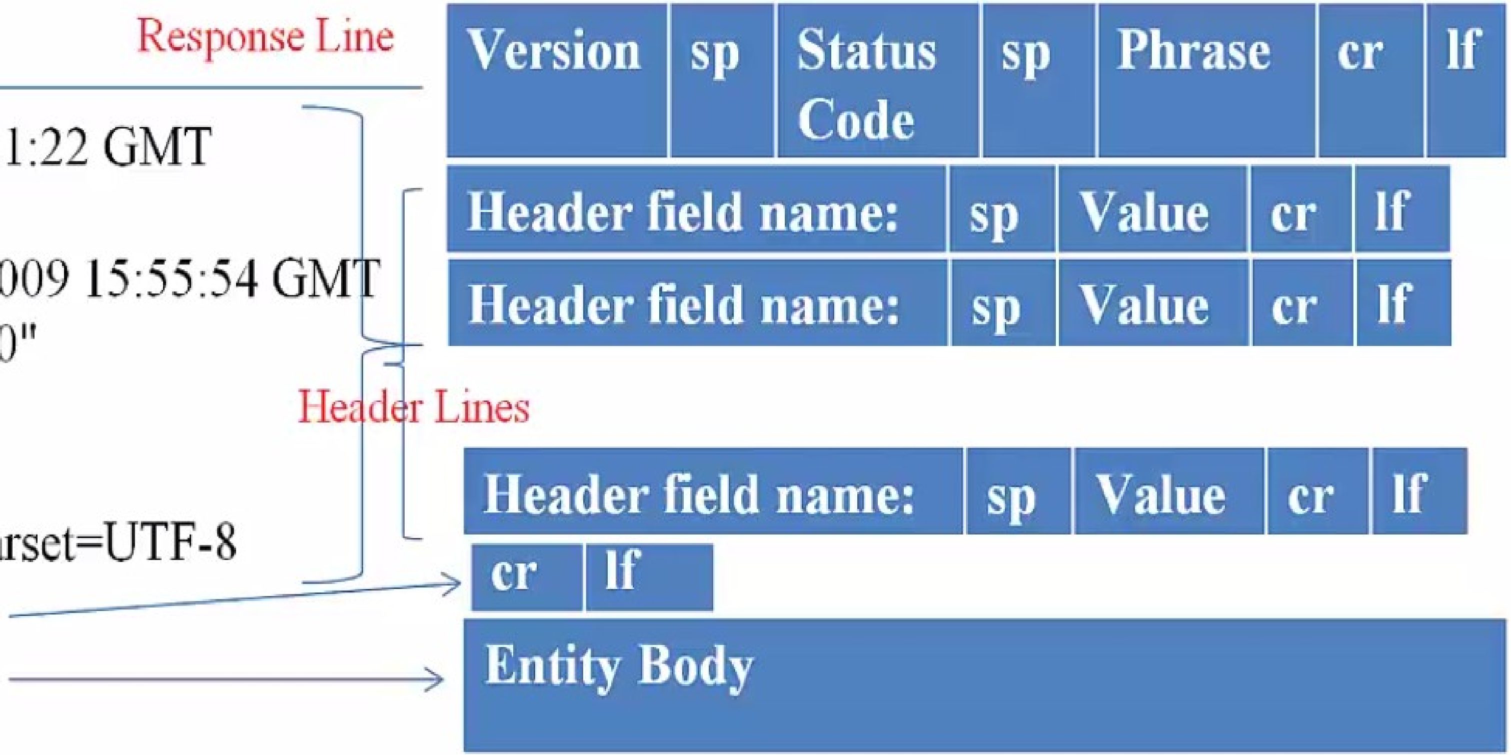
Methods

Method	Description
GET	Request for a web object
HEAD	Request for header fields (no body); Useful for debugging, get time of last modification
PUT	Upload an object to a specified path on a web server; body of request contains the object; used with web publishing tools
POST	Similar to PUT, except that object contained in body is "appended"; often used when user fills forms;
DELETE	Remove the object
TRACE	Asks server to echo incoming request; Useful for debugging
CONNECT	Used to facilitate secure connection when using Proxy servers
OPTIONS	Query server about its properties or that of an object
PATCH	Similar to PUT except permits partial modification to an object instead of a full replacement

Response Message Format

```
HTTP/1.1 200 OK
Date: Sat, 02 Nov 2013 12:31:22 GMT
Server: Apache
Last-Modified: Fri, 02 Jan 2009 15:55:54 GMT
ETag: "424c001-fd-f9d8d680"
Accept-Ranges: bytes
Content-Length: 253
Content-Type: text/html; charset=UTF-8

data data data data data ...
```



Version | sp | Status Code | sp | Phrase | cr | lf

Header field name: | sp | Value | cr | lf

Header field name: | sp | Value | cr | lf

Header field name: | sp | Value | cr | lf

cr | lf

Entity Body

Sample Status Code and Phrases

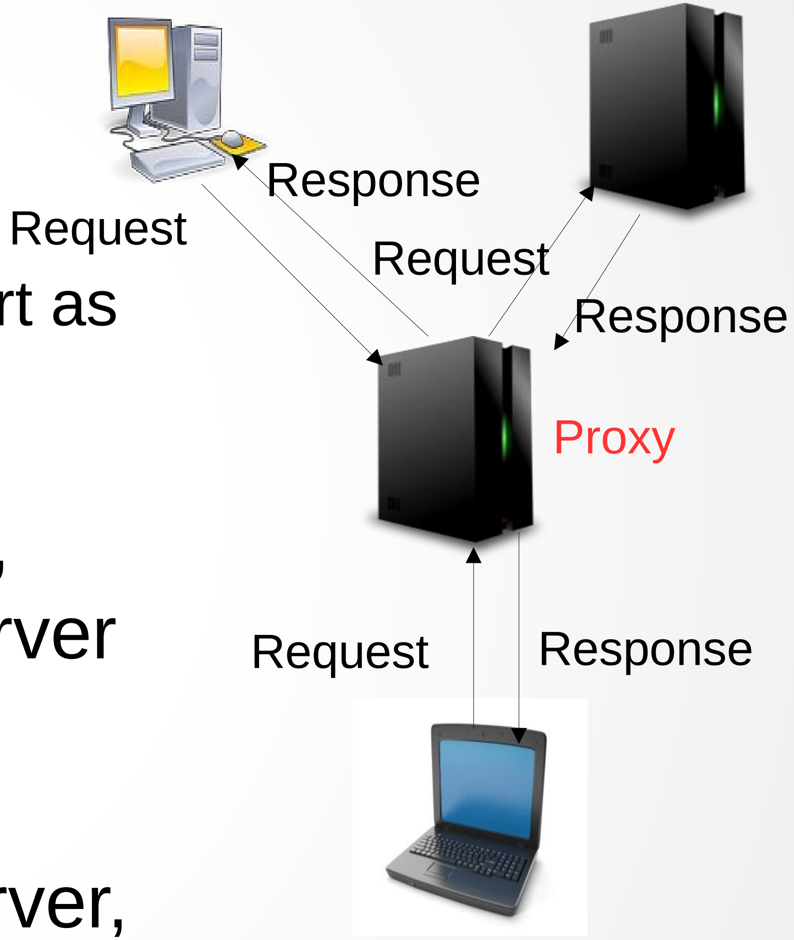
Status Code	Phrase	Description
200	OK	Request successful, information enclosed
301	Moved Permanently	Object moved; new url under Location:
400	Bad Request	Request could not be understood
404	Not Found	Requested object not found on server
503	Service Unavailable	Server is currently unavailable (overloaded)
505	HTTP Version not supported	Server does not support the HTTP version

Web Caches (Proxy Server)

- Browsers access web via a cache (proxy server)
  - – Can specify proxy address and port as part of browser's network settings

- If HTTP object request in cache, proxy returns the object (origin server not involved)

- Else, Proxy contacts original server, obtains object and returns it to client



Proxy Server

- Proxy server installed by ISP
  - Universities, Companies, local ISPs

- Proxy acts as server to clients and as client to external servers

- Internet dense with caches

Advantages

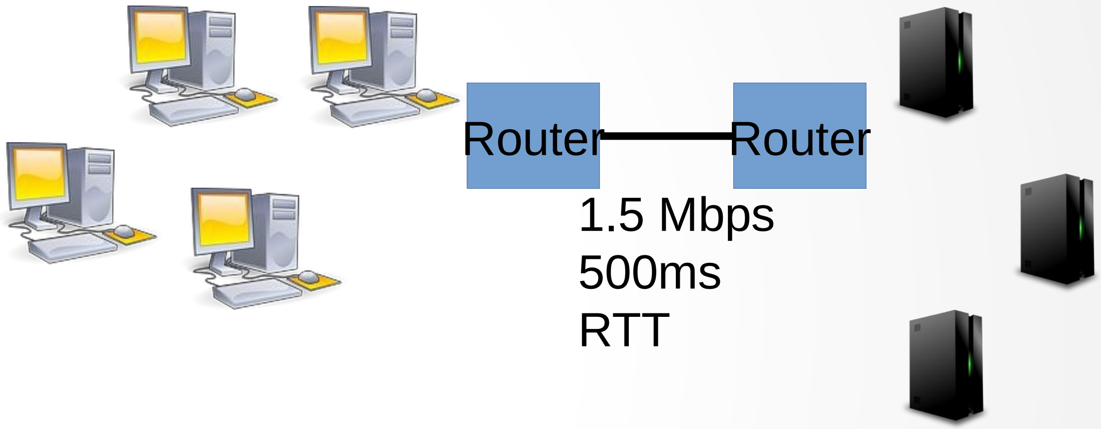
- Reduces response time of client request
- Saves bandwidth (prevents downloading of same content multiple times)
- Helps block undesired sites etc
- Enables “poor” content providers to effectively deliver content

Example

- Assumptions:

– Average object size is 100Kbits

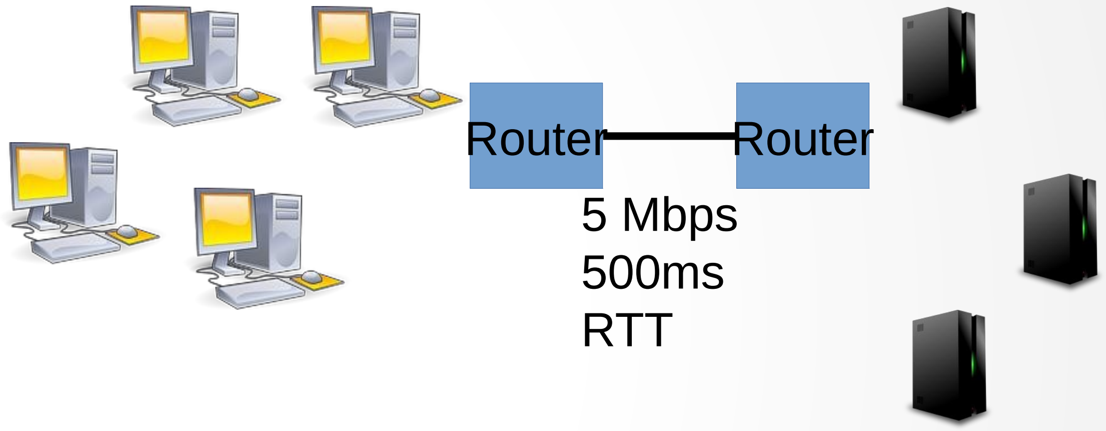
– Request rate inside organization is 20 requests/sec



- Approximately, what is the average response time?
    - Order of minutes (governed by queuing delay; considering TCP handshake time)

High Bandwidth Link

- Assumptions:
  - – Average object size is 100Kbits
  - – Request rate inside organization is 20 requests/sec



- Approximately, what is the average response time?
  - Around 1 sec but can be a costly upgrade

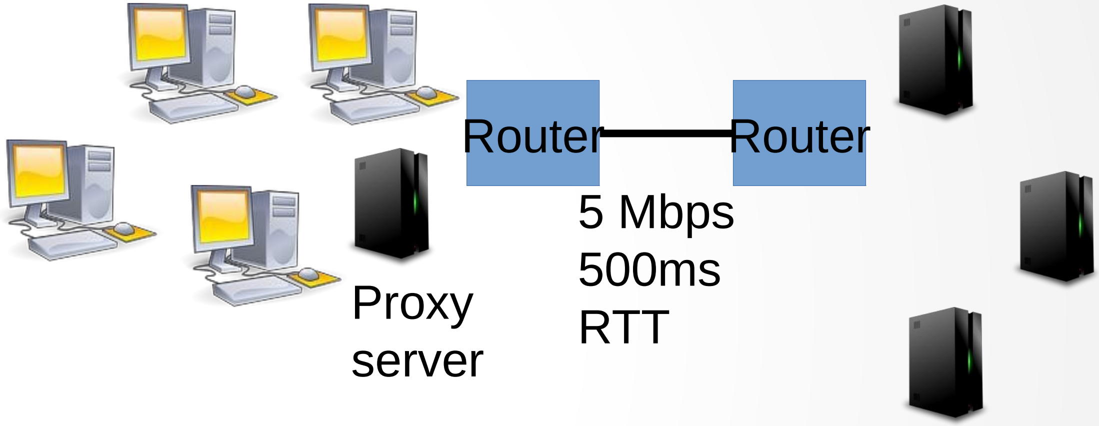
Proxy Server

- Assumptions:

– Average object size is 100Kbits

– Request rate Inside organization Is 20 requests/sec

– Hit rate of 0.4



- Approximately, what is the average response time?
   - Roughly 1 sec [60% requests take ~ 1 sec, while 40% take a few ms]

Stale Cache

- Problem with Proxy Server
  - How to verify if object in cache is up to date?
- HTTP 1.1 provides means to do this via cache-control headers
  – max-age=[seconds] : indicates freshness of the object

  – public : content is cacheable

  – private : content is not cacheable at proxy (can be cached at browser)

  – no-cache : need validation before releasing content

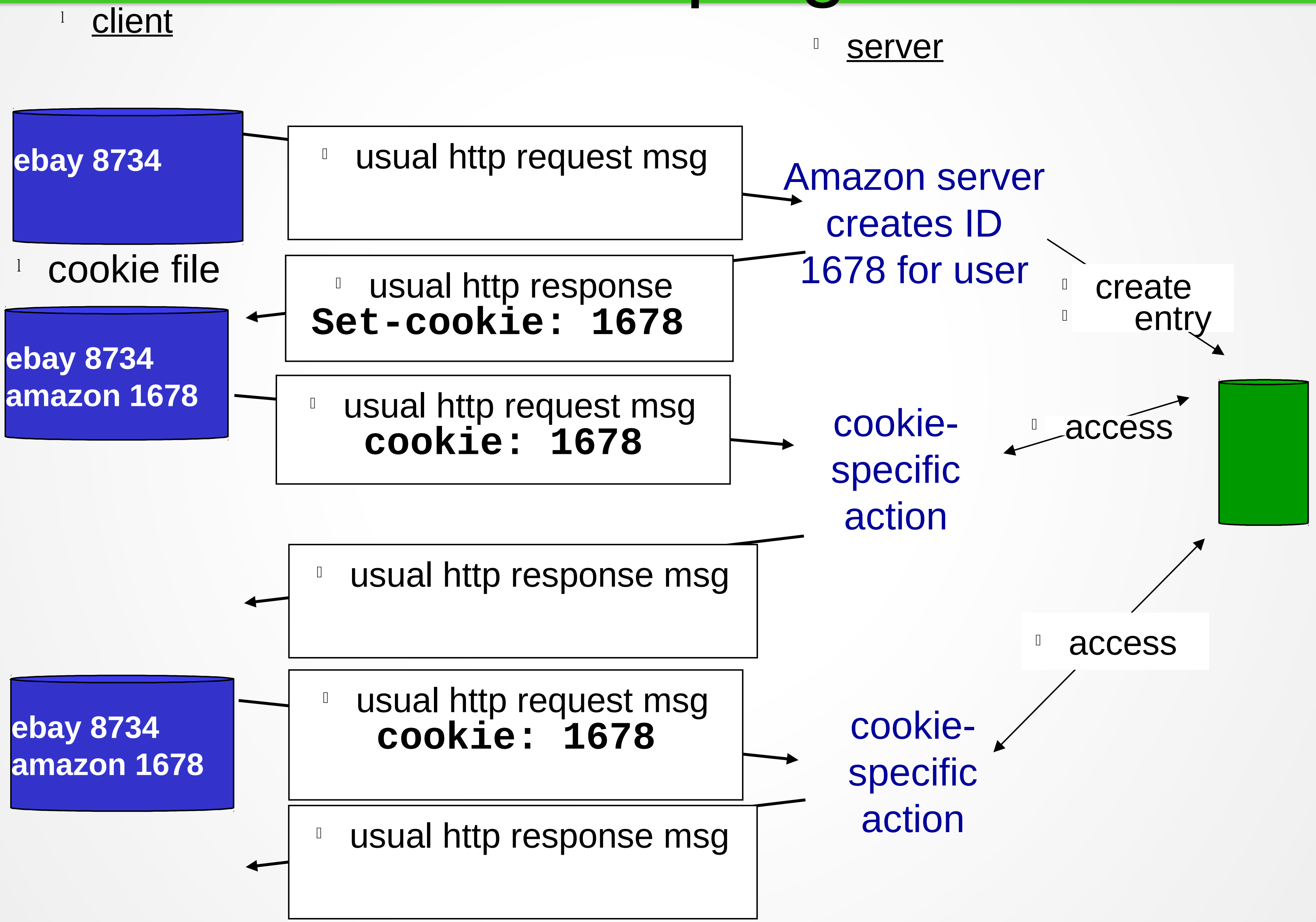
  – no-store : do not cache under any condition

  – E.g. Cache-Control: max-age=3600, public

Cookies

- HTTP is a stateless protocol, but servers like to maintain state specific to a user
    - E.g. shopping cart info; user’s preference settings; browsing activity;
- Solution: Cookies
    - Server sends ‘small amount of data (cookie)’ to store at client
    - Every time client contacts the server (same domain), browser sends cookie (of the domain) to server

Cookies: Keeping “state”



client

server

ebay 8734

usual http request msg

Amazon server creates ID 1678 for user

cookie file

usual http response
`Set-cookie: 1678`

create
entry

ebay 8734
amazon 1678

usual http request msg
`cookie: 1678`

cookie-specific action

access

usual http response msg

access

ebay 8734
amazon 1678

usual http request msg
`cookie: 1678`

cookie-specific action

usual http response msg

Cookie Structure

Browsers need to support at least 4KB cookie size

- Name of cookie
- Value of cookie
- Expiry of cookie
- Path the cookie is good for
- Domain of the cookie
- Type of connection needed

Demo

Preferences → Privacy → History section → use custom setting for history → show Cookies