Computer Networks
Chapter 2.2

Jong-won Lee

Handong University

User-server state: cookies

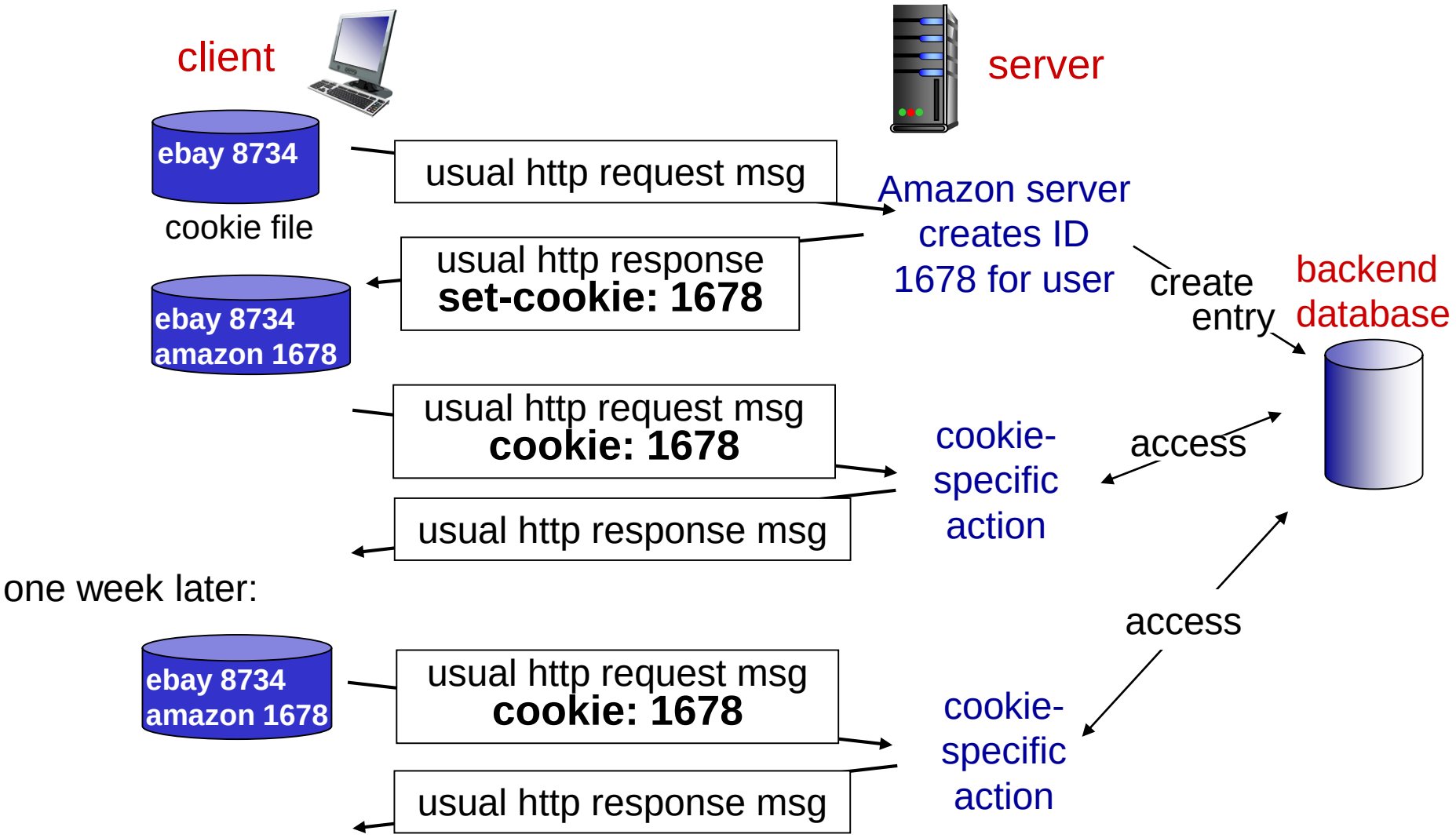
- HTTP is a stateless protocol
  - Server forgets about each client as soon as it sends response.
- Issues to stateless behavior
  - When a Web site wants to identify users
  - When the server wishes to restrict user access
  - When the server wants to serve content as a function of the user identity
- Cookie technology
  - When a server identifies a new user, it adds header to its response, containing an identifier for that user.
  - The client is expected to from the Set-Cookie header on its disk, and made to the same server.

User-server state: cookies

- A cookie is a short piece of data, not an executable code, and can not directly harm the machine.
- Four components of cookie
  1) line in the HTTP
  2) line in HTTP
  3) cookie file kept on user's host and managed by user's browser
  4) back-end database at Web site
- Problem in privacy

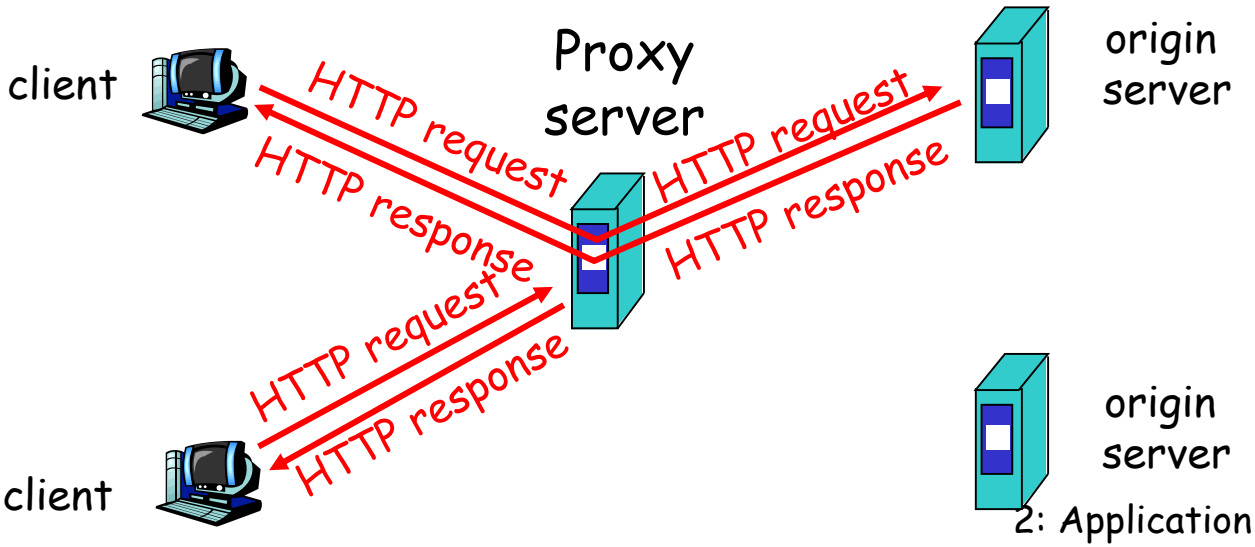
Cookies: keeping "state" (cont.)

Example: interaction with Amazon



Web caches (proxy server)

- Web cache/Proxy server
  - An intermediary entity that satisfies HTTP requests on the behalf of an origin Web server
  - User browsers must be configured so that all requests are first directed to its Web cache
    - object in cache: cache returns object
    - Else cache requests object from origin server, then returns object to client



Web caching

- Advantages
  - Reduce response time for client request.
  - Reduce traffic on an institution's access link.
- Disadvantages
  - Lower performance for objects that are not cached.

Caching example

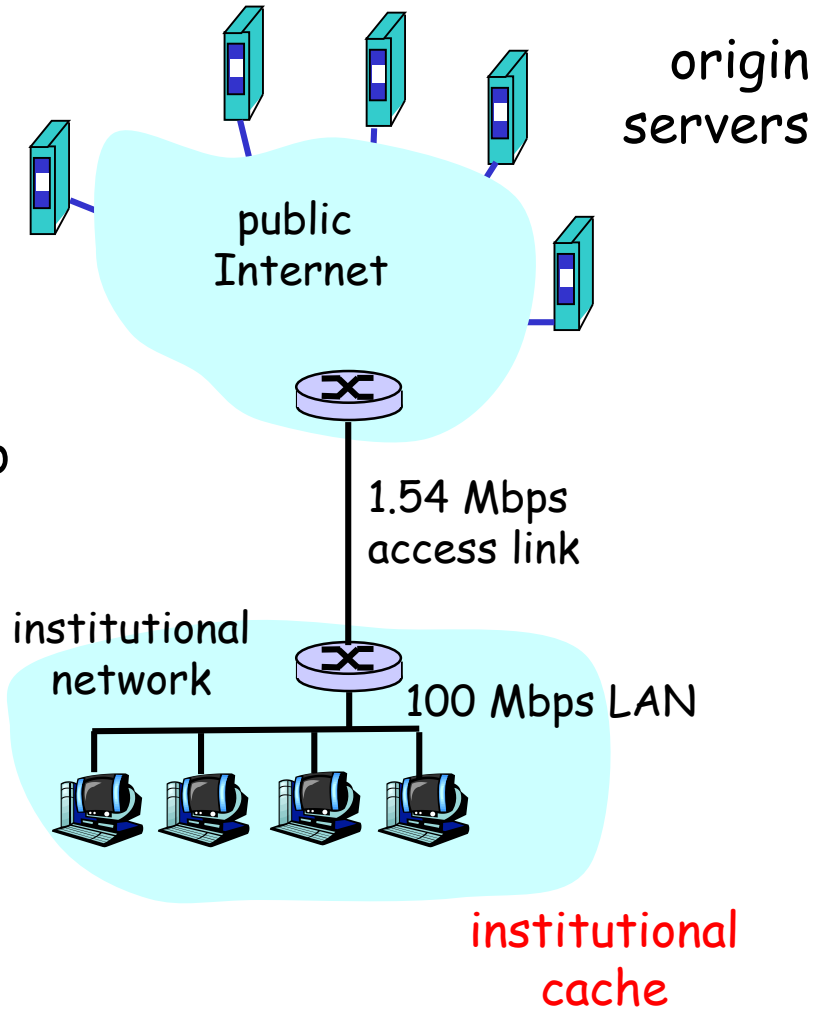
Assumptions

- average object size = 100k bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from ISP access router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 1.5%
- utilization on access link = 99%
- total delay = Internet delay + access delay + LAN delay

  = 2 sec + minutes + milliseconds



Caching example (cont)

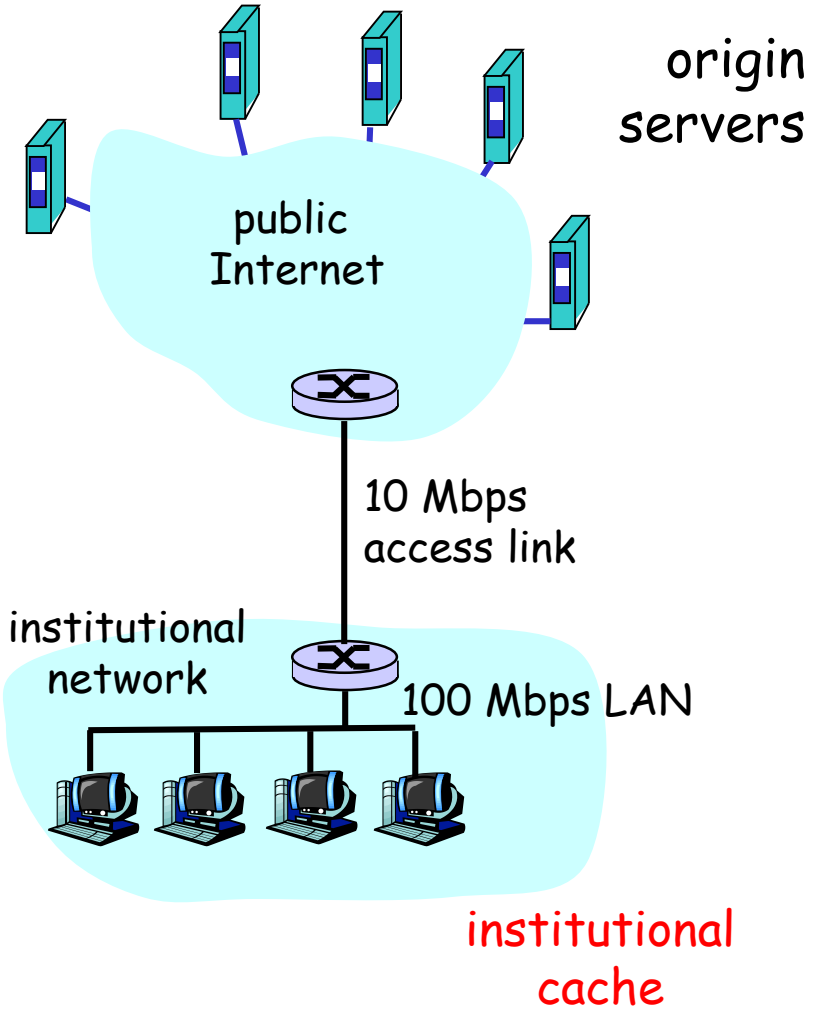
Possible solution

- increase bandwidth of access link to, say, 10 Mbps

Consequences

- utilization on LAN = 1.5%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs
- often a costly upgrade



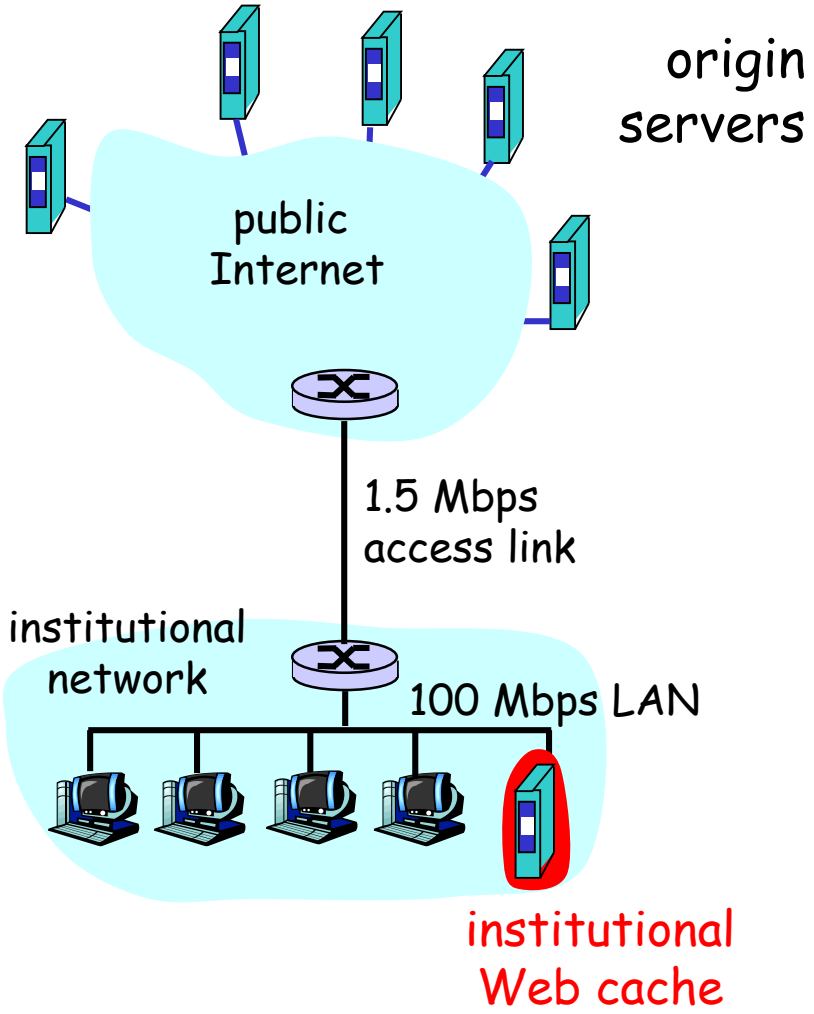
Caching example (cont)

Install cache

- suppose hit rate is .4

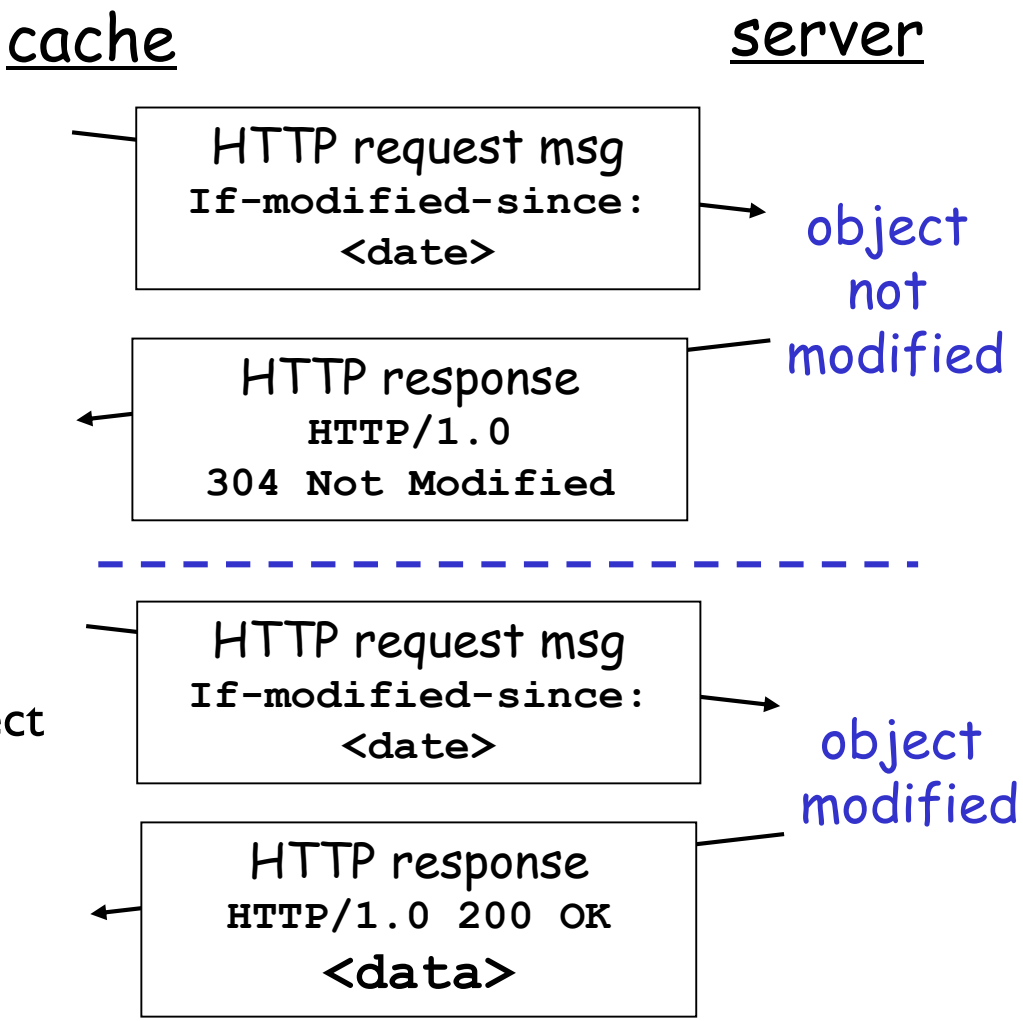
Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = 0.4*LAN access delay + 0.6*WAN access delay = 0.4*(~msec) + .6*(2+0.01) =~ 1.2 secs



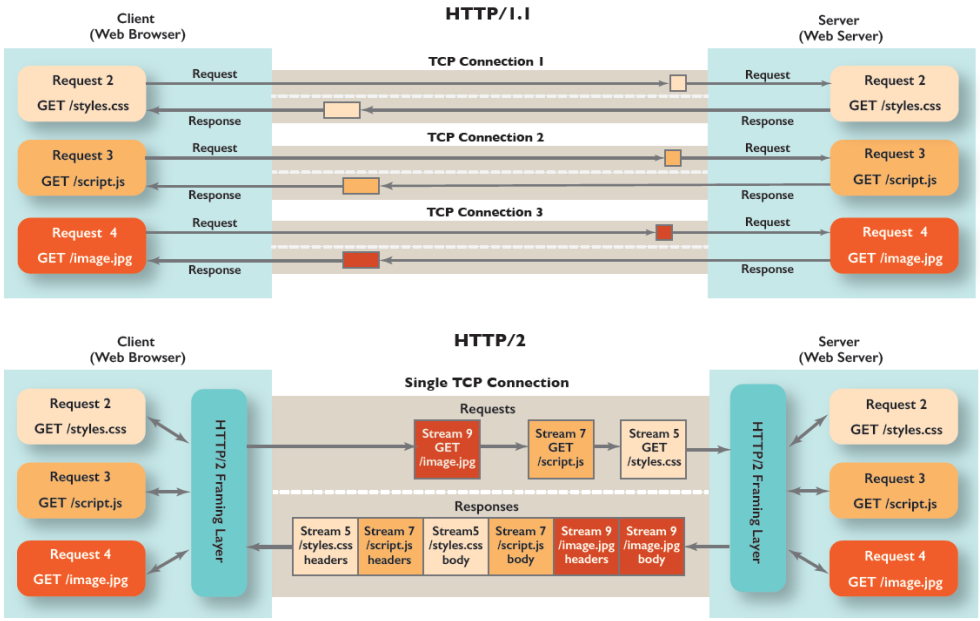
Web Cache Challenge

- Problem: an object in the cache might be
- Goal: don't send object if cache has old version
- Solution:
  - Use header line
  - Cache will include requested object in response only if object has been modified since the specified date



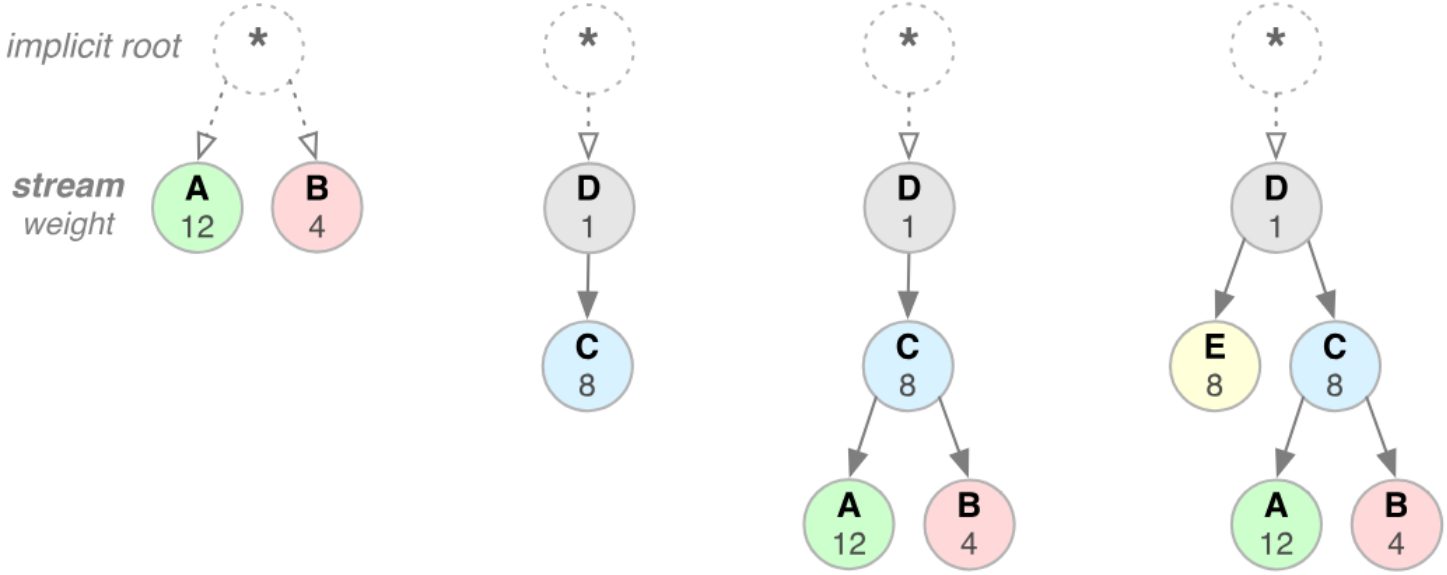
HTTP/2

- Features
  - HEADER frame + DATA frame
    - Binary format
  - Multiplexing



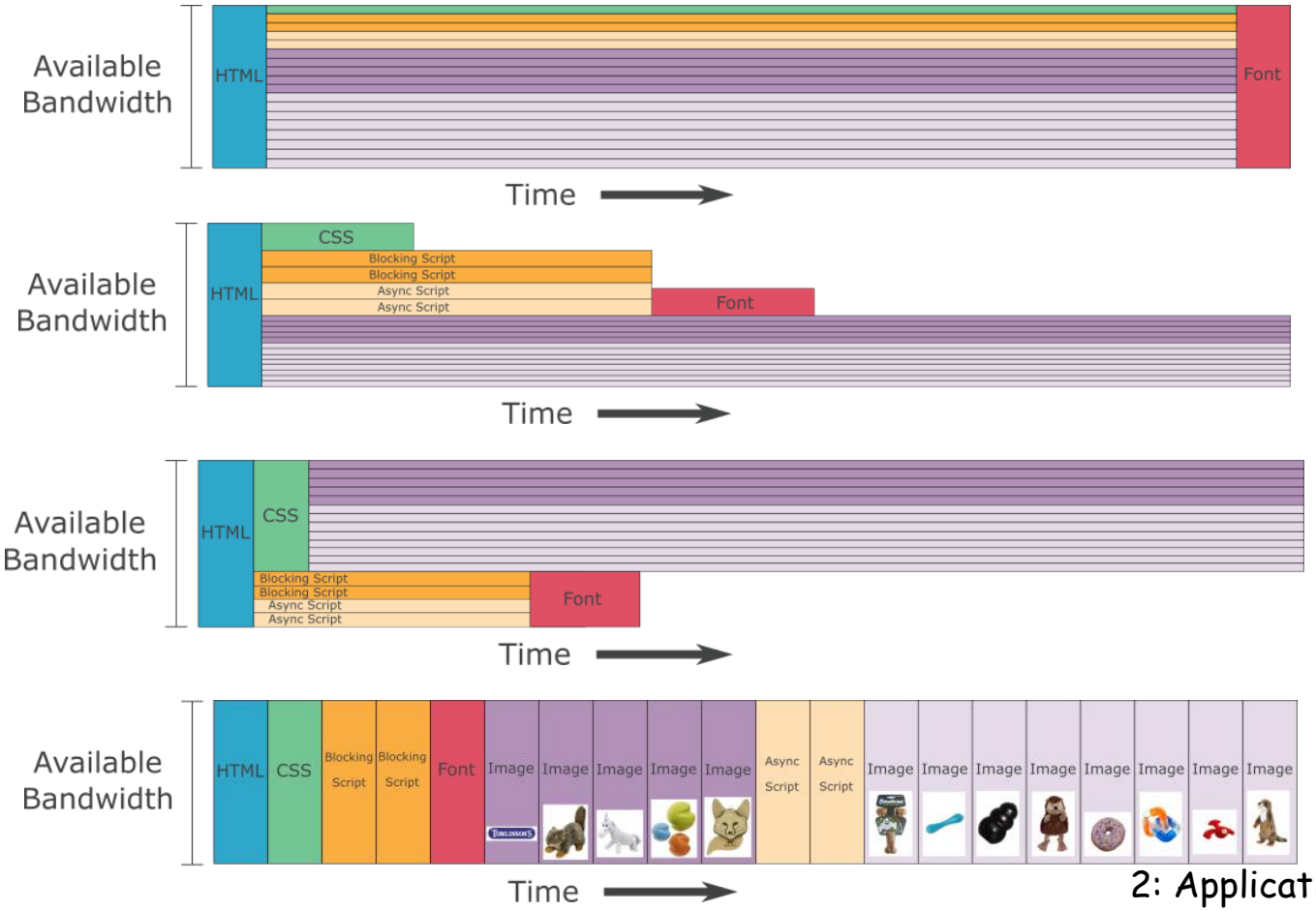
HTTP/2

- Features
  - Stream prioritization



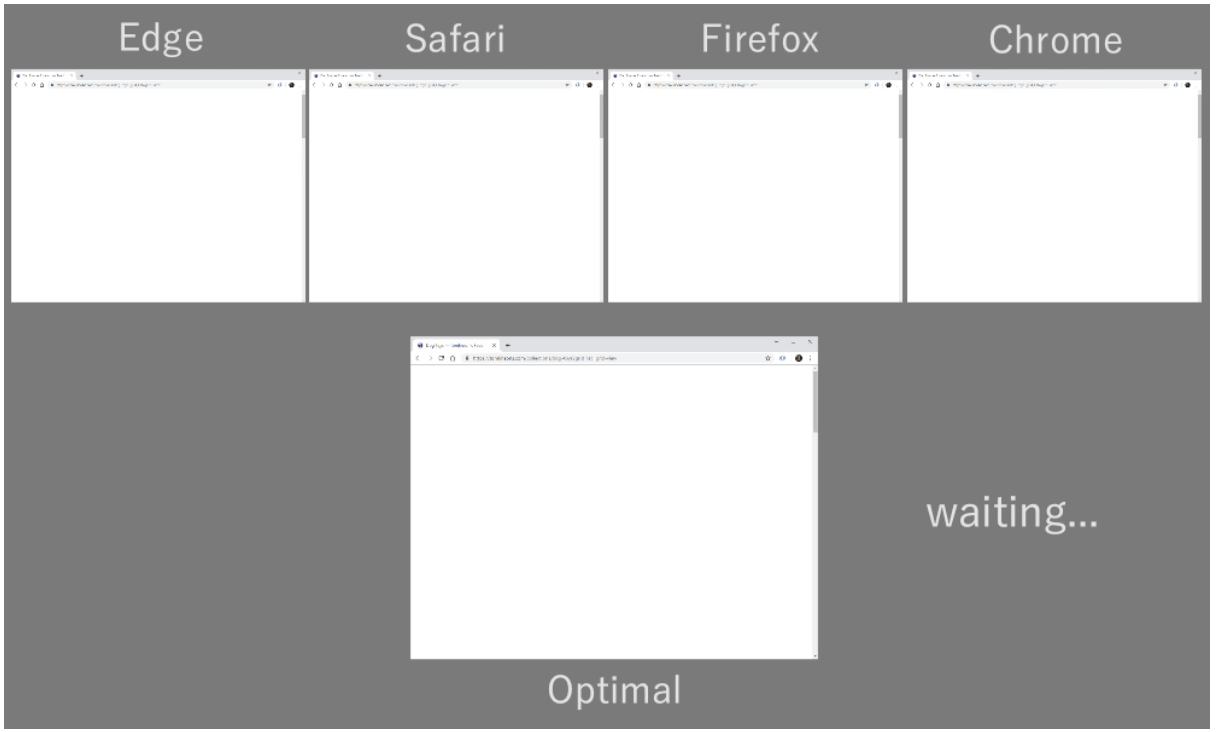
HTTP/2

- Stream prioritization
  - https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/



HTTP/2

- Stream prioritization
  - https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/



HTTP/2

- Features
  - Server push

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

HTTP/2

- Features
  - Server push

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

HTTP/2

- Features
  - Server push

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

HTTP/2

- Features
  - Flow control
    - Stream- and connection-based flow control
  - Header compression
    - HPACK compression mechanism
      - Huffman coding & index based