Application Layer (HTTP)

Lecture 3 | Part 2 of 2 | CSE421 – Computer Networks

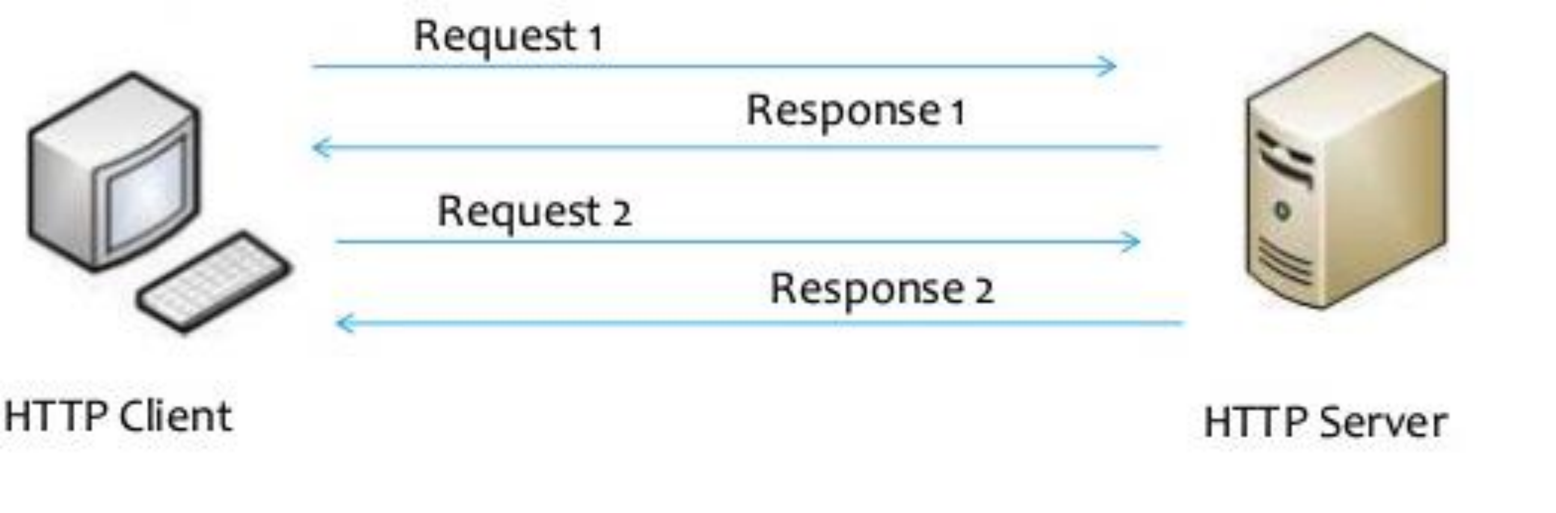
Department of Computer Science and Engineering
School of Data & Science

COOKIES

Objectives Part 4

- Stateless HTTP
- Cookies
- Example of how cookies operate
- Issues related to cookies

Stateless HTTP



- Do not remember previous request response chain.

HTTP is "stateless"

- Server maintains no information about past client requests

Protocols that maintain "state" are complex!

- Past history (state) must be maintained
- If server/client crashes, their views of "state" may be inconsistent, must be reconciled

User-server state: cookies

Many Websites use cookies

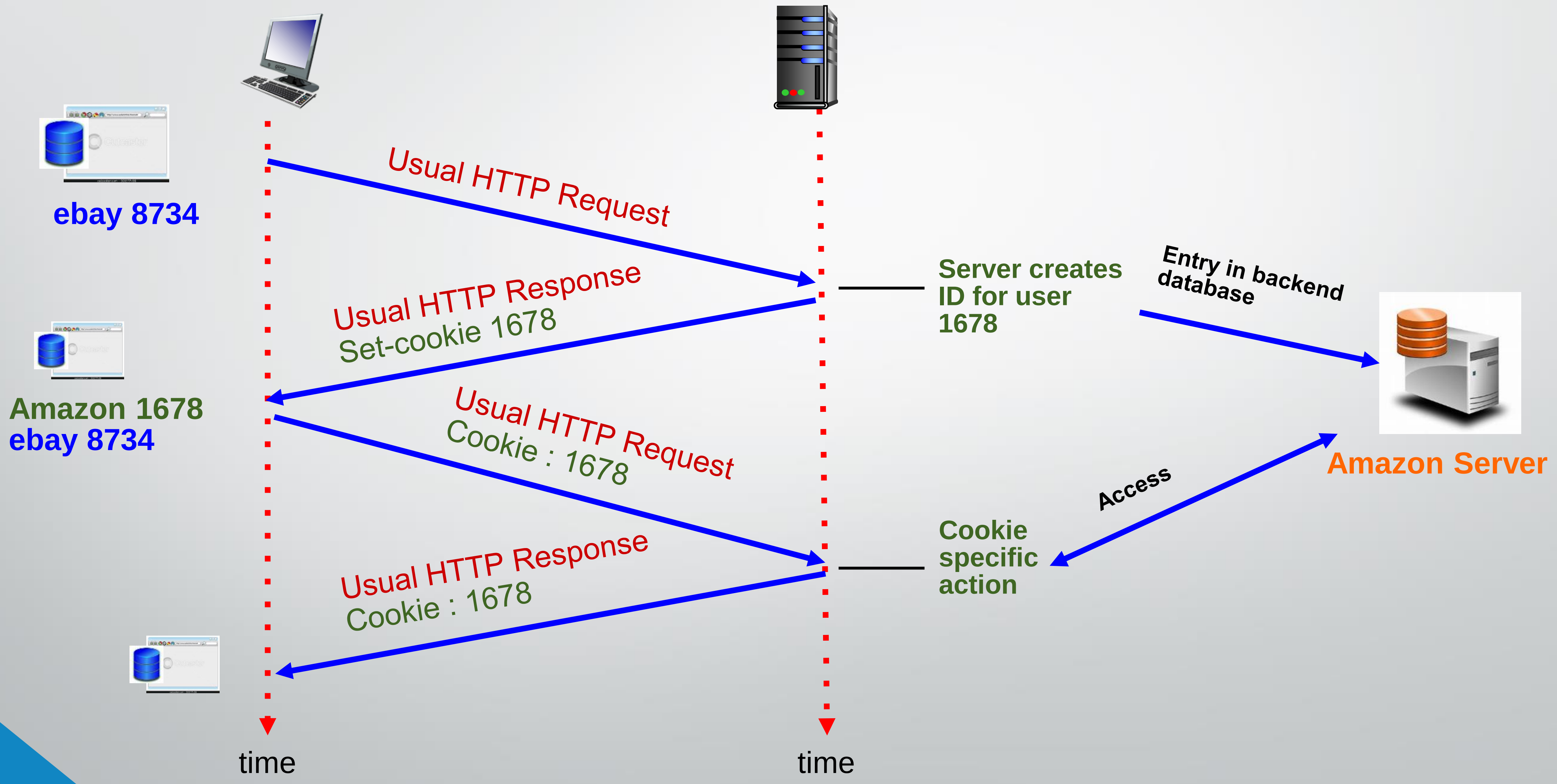
Four components:

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

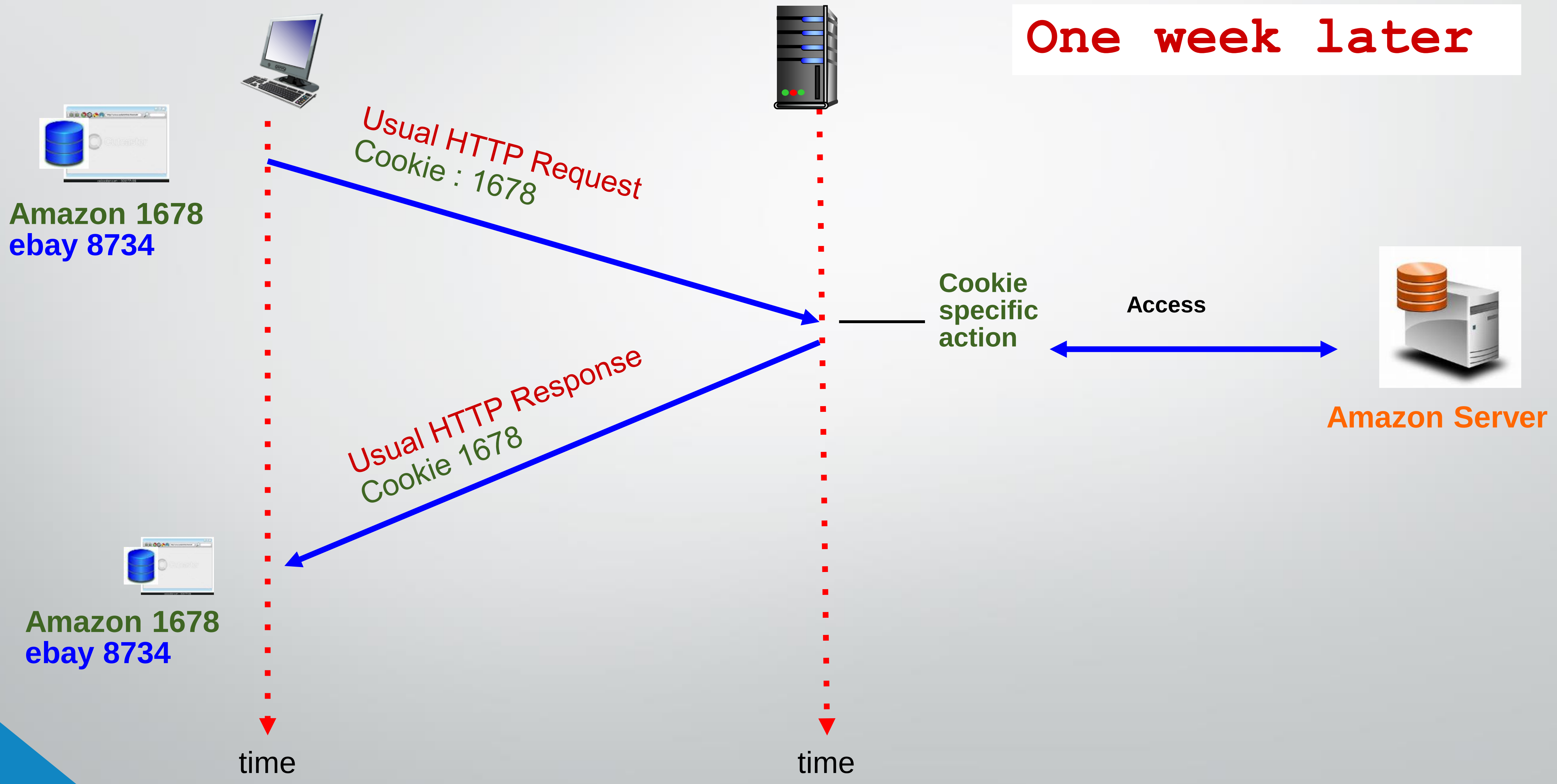
Example:

- Susan always access Internet from PC
- 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: keeping "state" (cont.)



Cookies: keeping “state” (cont.)



Cookies (continued)

What cookies can be used for:

- Authorization
- Shopping carts
- Recommendations
- User session state (Web email)

aside

Cookies and privacy:

- Cookies permit sites to learn a lot about you
- You may supply name and email to sites

How to keep "state":

- Protocol endpoints: maintain state at sender/receiver over multiple transactions
- Cookies: http messages carry state

Web Cache or Proxy Server

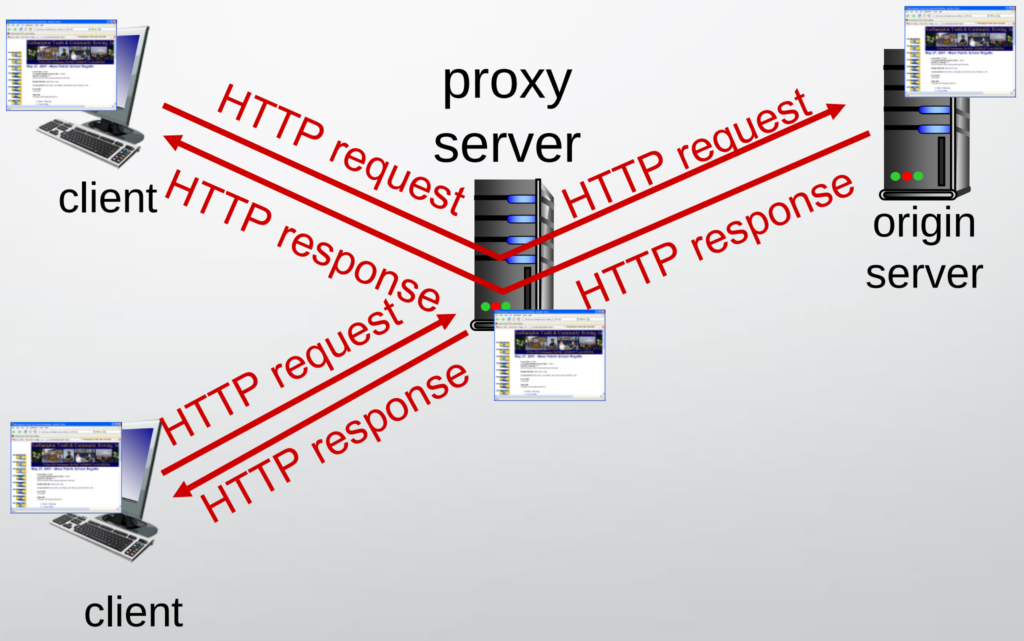
Objectives – Part 5

- What is a Web Cache (Proxy Servers)?
- Advantages of Web Cache
- Web Caching Example
- Problems of Web Caching
- Conditional-GET

Web caches (proxy server)

Goal: Satisfy client request without involving origin server

- User sets 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



More about Web caching

- **A Proxy Server acts as both client and server**
  - Server for original requesting client
  - Client to origin server
- **Typically Proxy Servers are installed by ISPs**
  - University
  - Company
  - Residential ISP

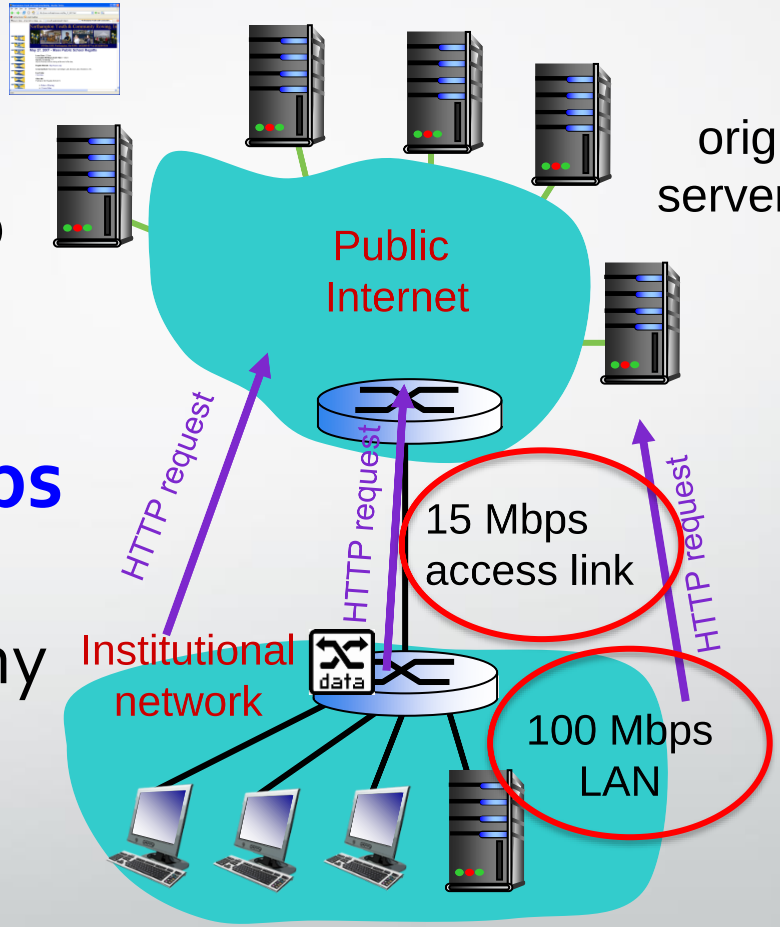
Advantages of Web Caching

- Reduce response time for client request
- Saves bandwidth (prevents downloading of same content multiple times)
- Helps log usage, block unwanted traffic
- Internet dense with caches:
  - enables "poor" content providers to effectively deliver content (so too does P2P file sharing)

Caching example:

Assumptions:

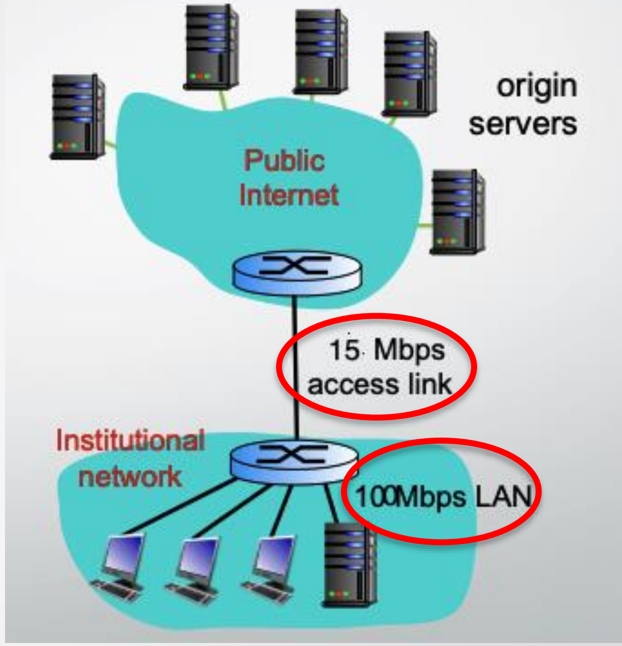
- Avg object size: **1 Mbits**
- Avg request rate from browsers to origin servers: **15/sec**
- Institutional Bandwidth: **100 Mbps**
- RTT from institutional router to any origin server: **2 sec**
- Access link rate: **15 Mbps**



Caching example:

Assumptions:

- Avg object size: 1 Mbits
- Avg request rate from browsers to origin servers:15/sec
- RTT from institutional router to any origin server : 2 sec



What is the average response time?

Average response time = LAN Delay + Access Delay + Internet Delay

Traffic Intensity on LAN = (Avg Req/sec * Avg Obj Size)/Transmission Link Bandwidth

=(15*1000000) / 100000000= 0.15

Traffic intensity on the Access Link = (15*1000000)/ (15*1000000)= 1

Internet Delay = 2 sec (RTT)

Consequences:

- LAN utilization: **15%**
- Access link utilization = **100%**

Caching example: Solution

What is the average response time?

Average response time = LAN Delay + Access Delay + Internet Delay

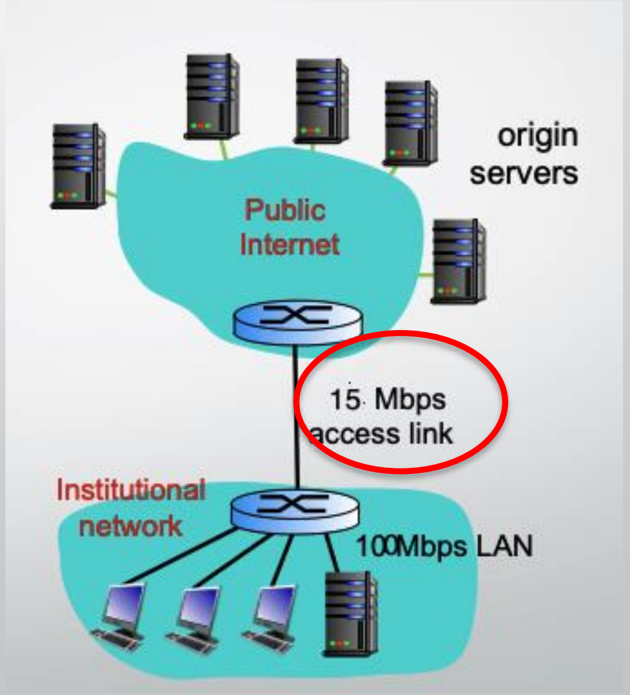
Consequences:

- LAN utilization: **15%**
- Access link utilization = **100%**

Delay is in tens of msecs

Delay is in of minutes.

Delay is 2 secs.



Solution of reducing delay:

1) Increase the bandwidth of the access link.

1) Install a web cache or proxy server at the institutional network.

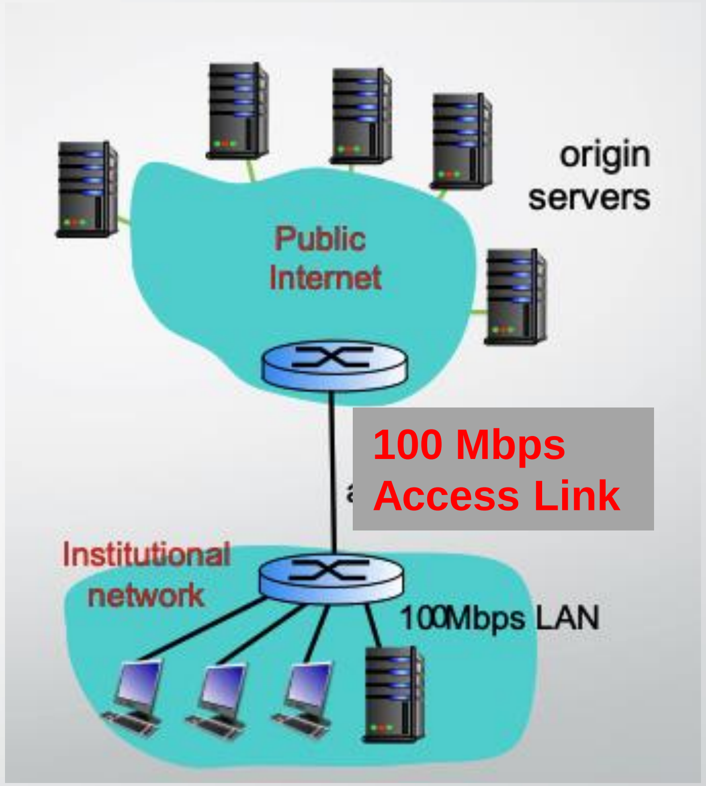
Caching example: Solution 1

Assumptions:

- Avg object size: 100K bits
- Avg request rate from browsers to origin servers:15/sec
- RTT from institutional router to any origin server: 2 sec
- Access link rate: ~~15 Mbps~~ 100 Mbps

Consequences:

- LAN utilization: 15%
- Access link utilization = ~~100%~~ 15%



Average response time = LAN Delay + Access Delay + Internet Delay

= msecs + ~~mins~~ msecs + 2 secs

LAN Delay and Access delay ***becomes negligible***

Cost: increased access link speed (not cheap!)

Caching example: Solution 2

Assumptions:

- Average hit rate: 40%

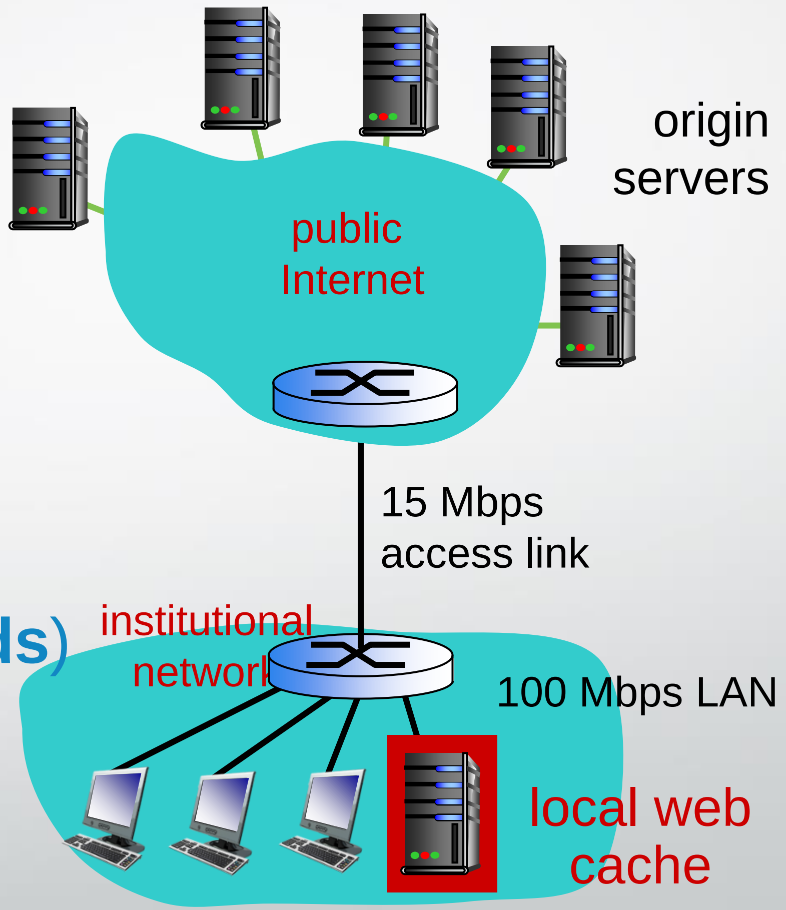
Consequences:

- **LAN delay** = within 10 milliseconds
- **Access delay** = tens of milliseconds in a 15 Mbps Link

- **Total Response time**:

= **0.4·(0.01 seconds)**+**0.6(0.01**+ **2 seconds)**

= **1.2 secs**



Cost: web cache (cheap!)

Stale Cache

- One problem of using Proxy server
  - The object housed in the Web server may have been modified since the copy was cached at the client.
- HTTP has a mechanism that allows a cache to verify that its objects are up to date.
- This mechanism is called the **conditional GET**.
- An HTTP request message is a so-called conditional GET message if
  1) The request message uses the GET method
  2) The request message includes an If-Modified-Since: header line.

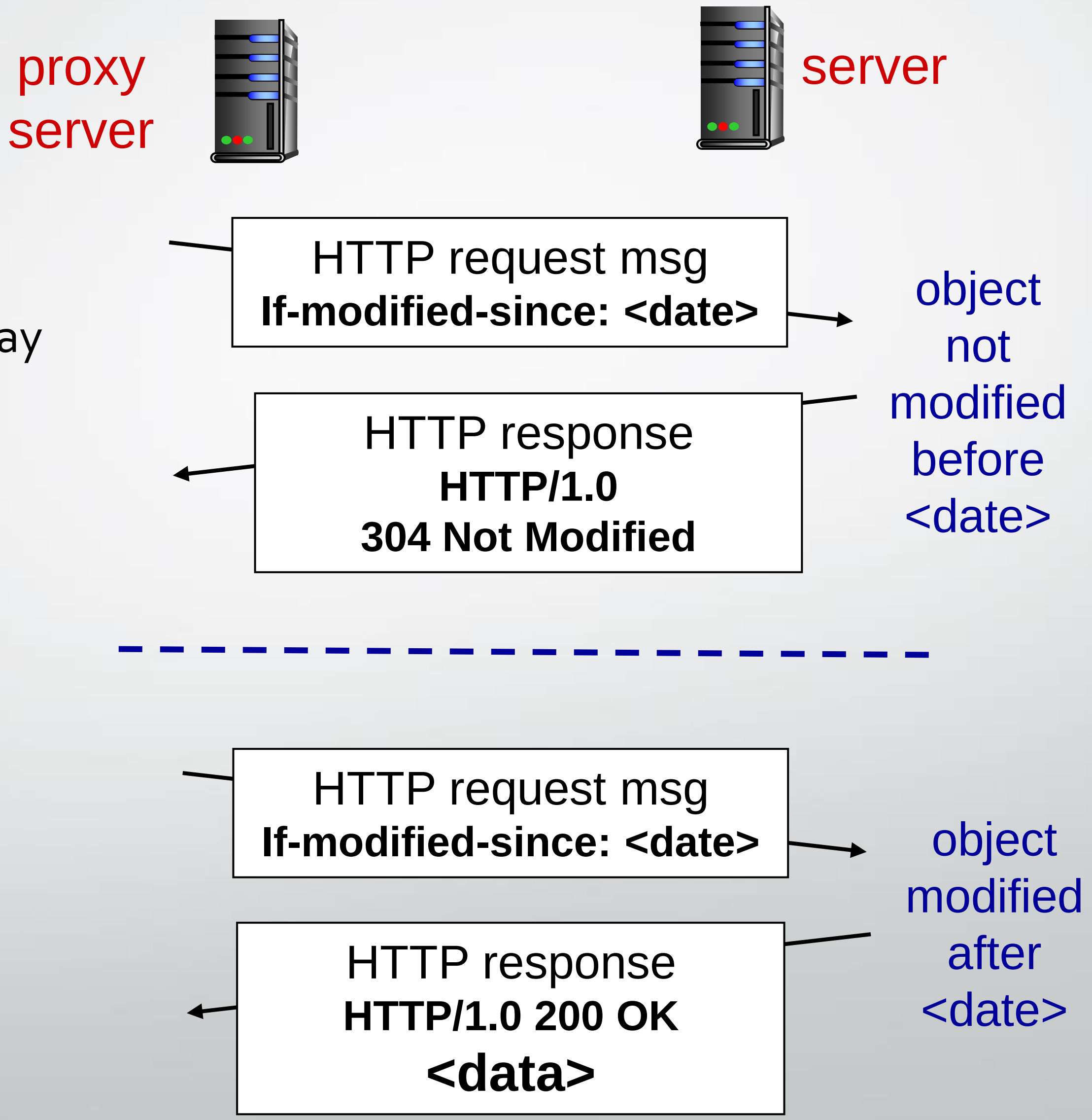
Conditional GET

- ***Goal:*** Don't send object if cache has up-to-date cached version
  - No object transmission delay
  - Lower link utilization
- ***Cache:*** specify date of cached copy in HTTP request

```
If-modified-since:
  <date>
```

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

```
HTTP/1.0 304 Not
  Modified
```



HTTP/2

Key goal: decreased delay in multi-object HTTP requests

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)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

Newer versions of HTTP

HTTP/2

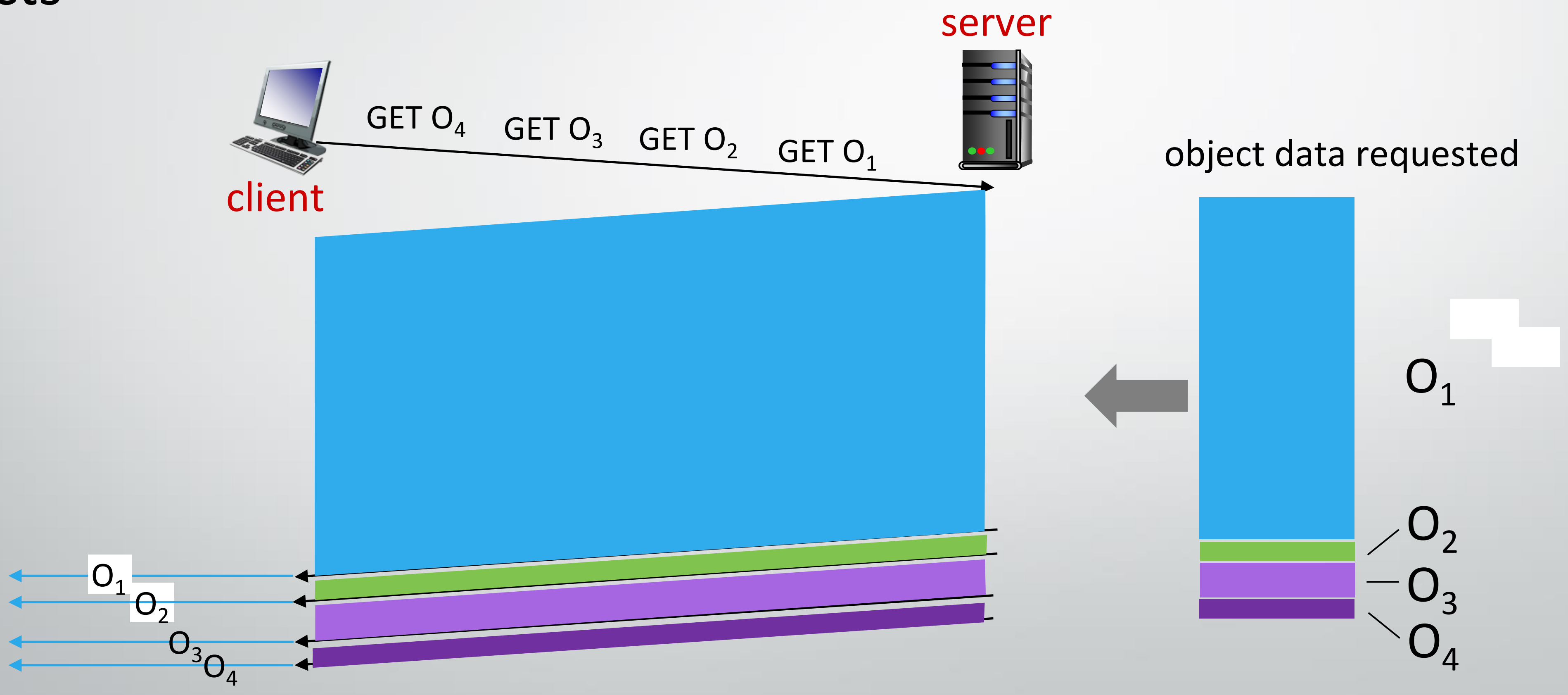
Key goal: decreased delay in multi-object HTTP requests

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

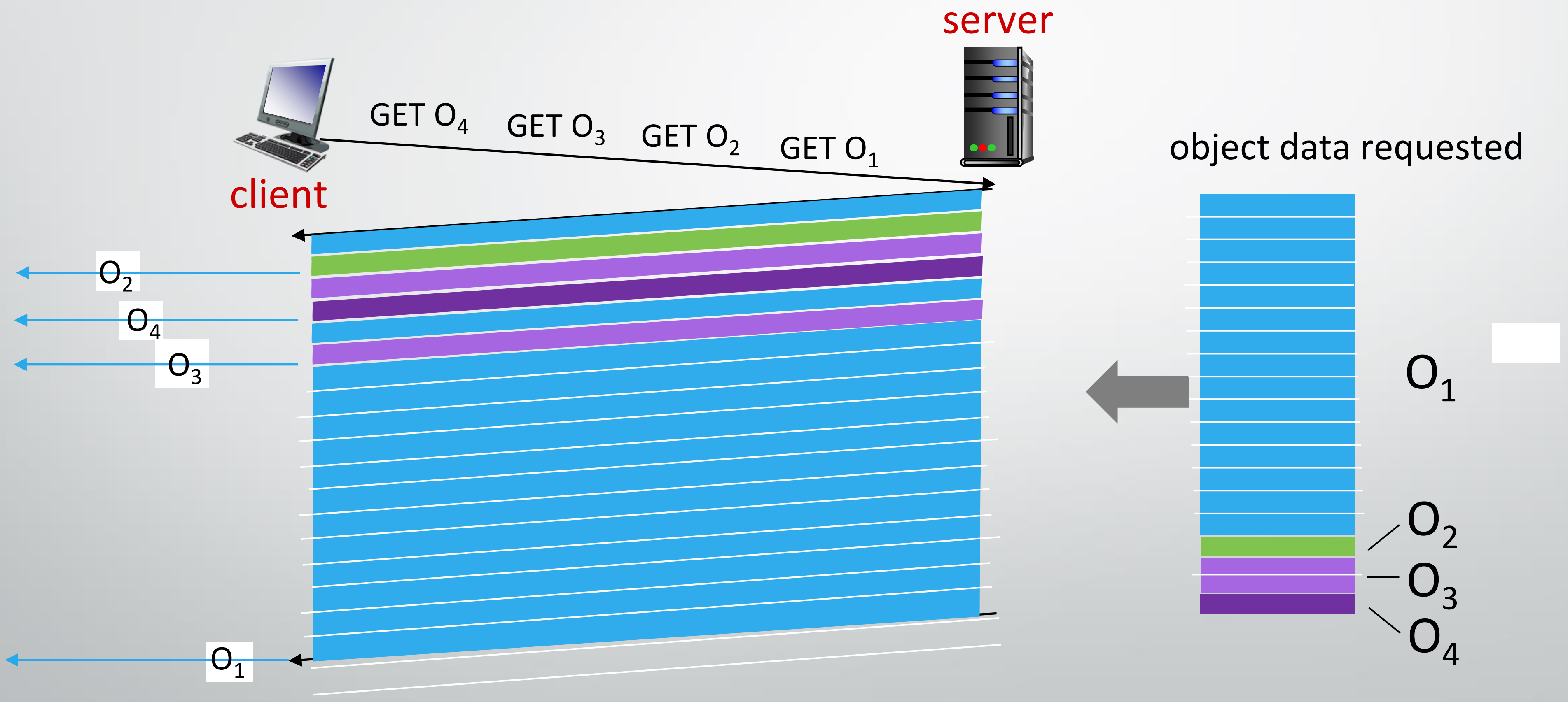
HTTP 1.1: client requests 1 large object (e.g., video file) and 3 smaller objects



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

HTTP/2: mitigating HOL blocking

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



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

HTTP/2 to HTTP/3

HTTP/2 over single TCP connection means:

- recovery from packet loss still stalls all object transmissions
  - as in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- no security over vanilla TCP connection
- HTTP/3: adds security, per object error- and congestion-control (more pipelining) over UDP
  - more on HTTP/3 in transport layer

YouTube Link:

https://www.youtube.com/watch?v=4M39gEPWPYs

THE END