HTTP: Hyper Text Transport Protocol

Web and HTTP

First, a review…

- *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.,

`www.someschool.edu/someDept/pic.gif`

host name | path name

How the Web works? (continued)

- in-class description

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- Used to request the URL and get the URL
- client/server model
  - *client:* browser issues GET requests (using HTTP protocol)
  - *server:* Web server sends (using HTTP protocol) objects in response to GET requests

HTTP request message

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.cs.stonybrook.edu\r\n
User-Agent: Firefox/3.6.10\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
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\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\r\n
Date: Sun, 26 Sept 2015 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 20 Sept 2015 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
Cache-control: max-age=0, no-cache
\r\n
data data data data data ...
```

The Keep-Alive timeout of the server takes preference over the timeout of the client.

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 msg

304 Not modified

  - request not modified, use the cached copy

400 Bad Request

  – request msg not understood by server

404 Not Found

  – requested document not found on this server

505 HTTP Version Not Supported

In general, what do you need to create a protocol?

- In class exercise.

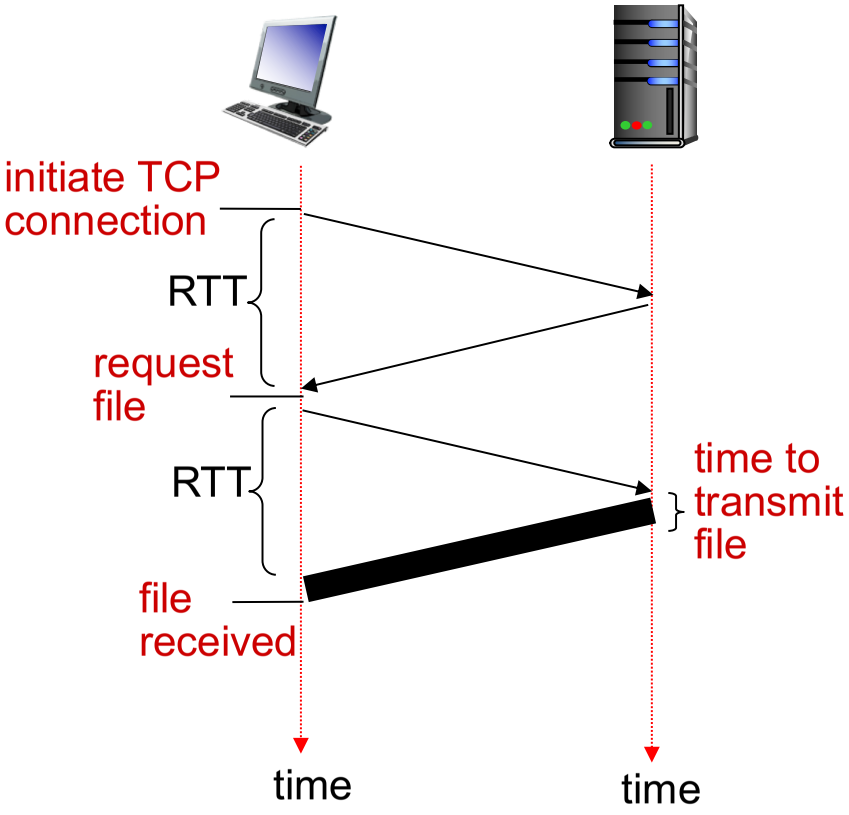
Response time estimation

RTT: time for a packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
  - This assumes HTTP GET piggy backed on the ACK
- file transmission time
- HTTP response time =

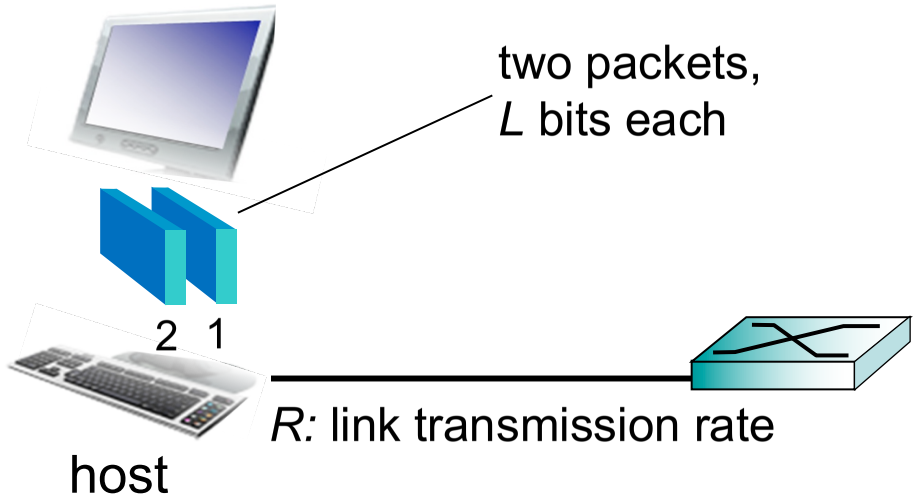
  2RTT+ file transmission time



RTT is an easy way to code transmission delay

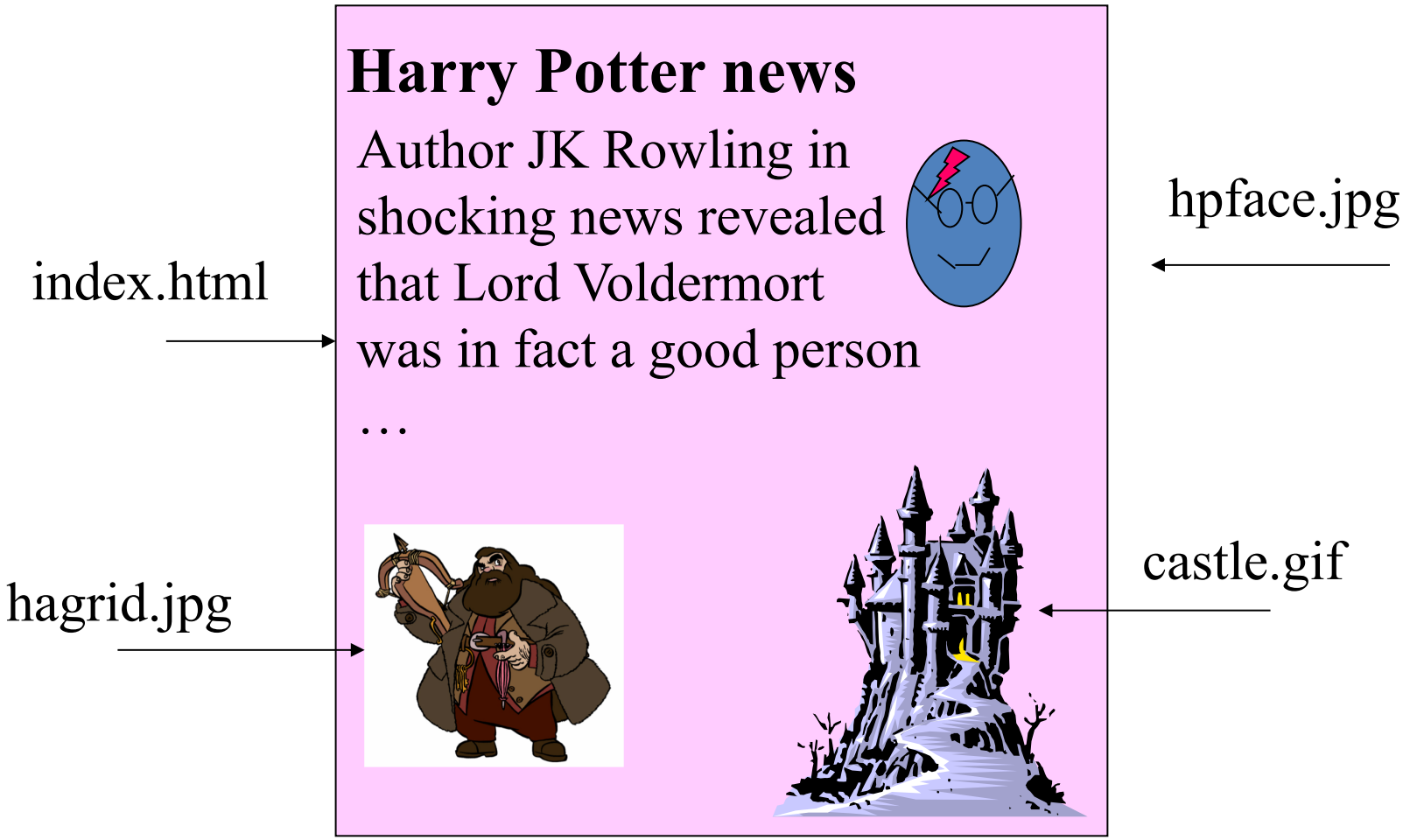
host sending function:

- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
  - link transmission rate, aka link *capacity, aka link bandwidth*

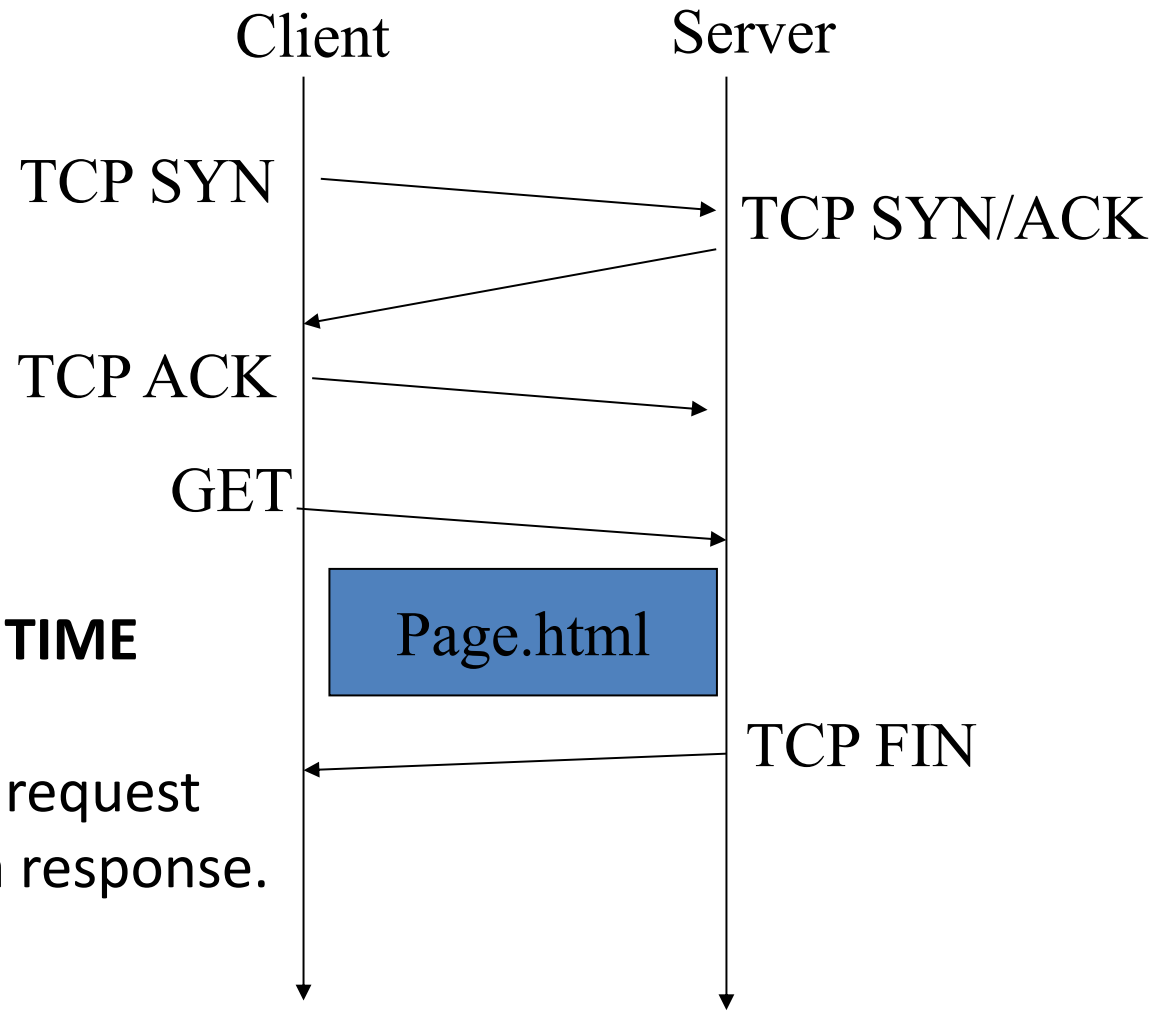


$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Example Web Page



HTTP Request/Response



ROUND TRIP TIME (RTT)

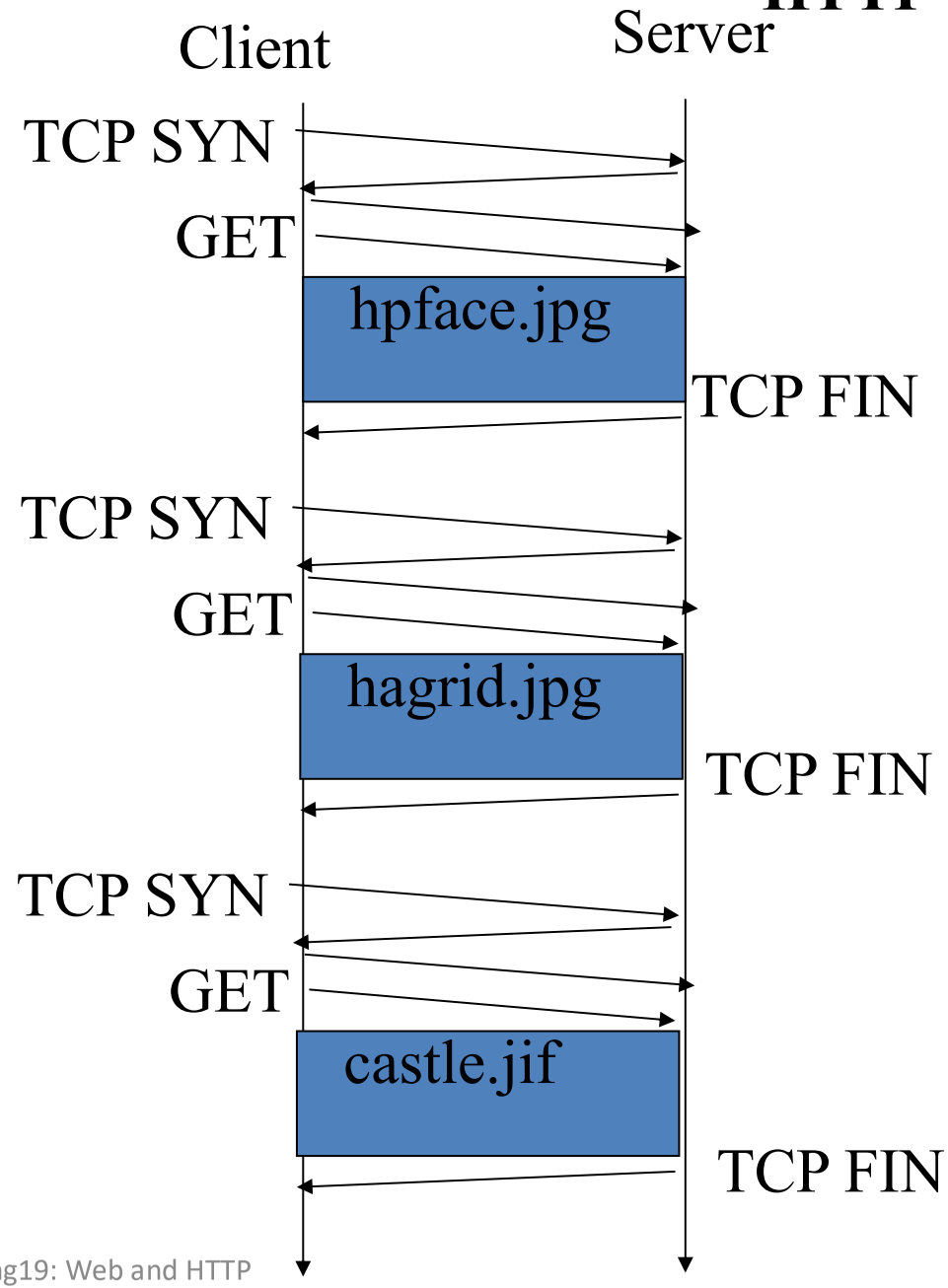
Time to send request

And receive a response.

Simple HTTP 1.0 (Non persistent HTTP)

Assume the html file is already transferred.

HTTP 1.0 req/response



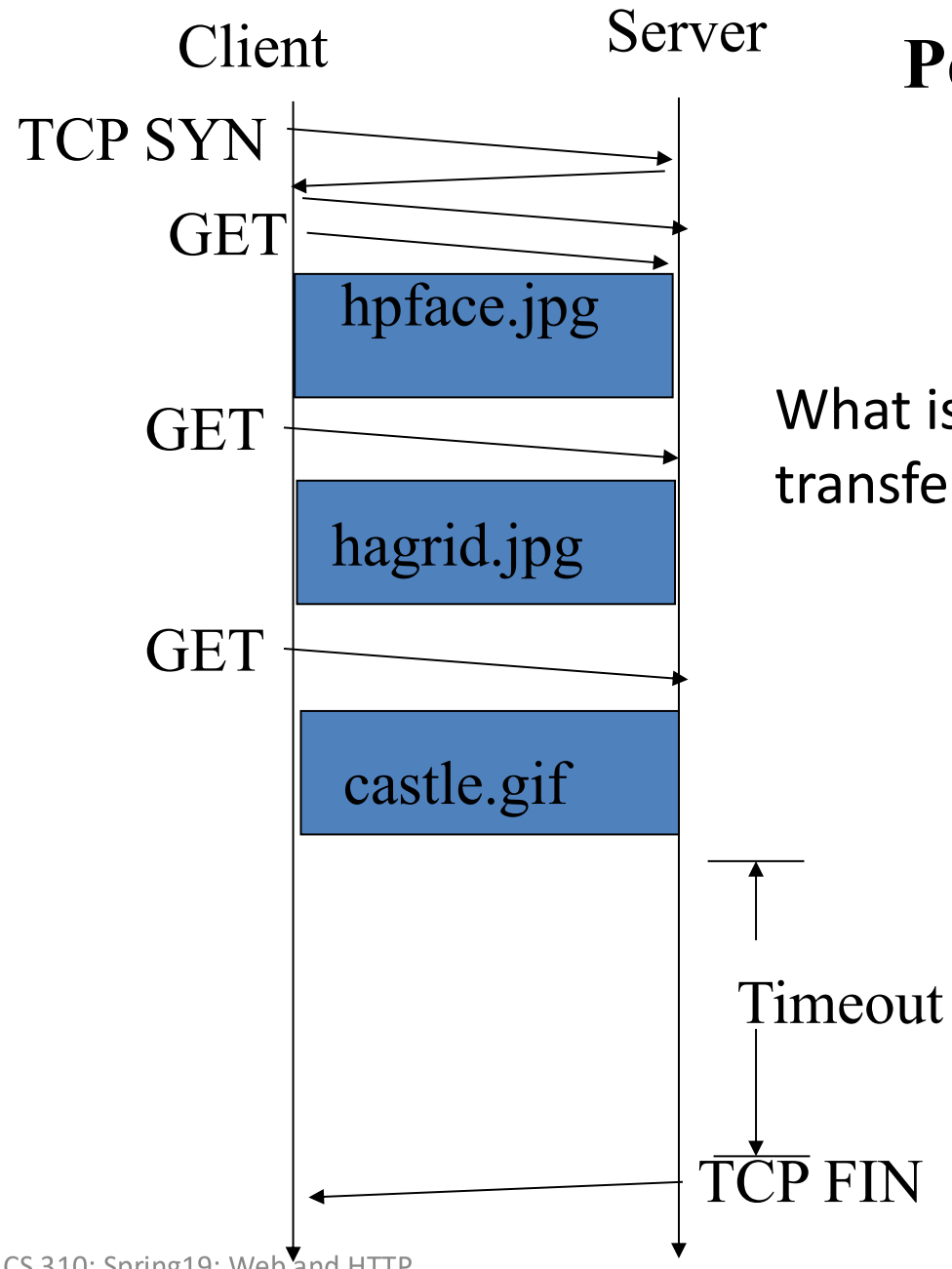
Non Persistent HTTP response time

- requires 2 RTTs + file transfer time per object
- OS overhead for *each* TCP connection

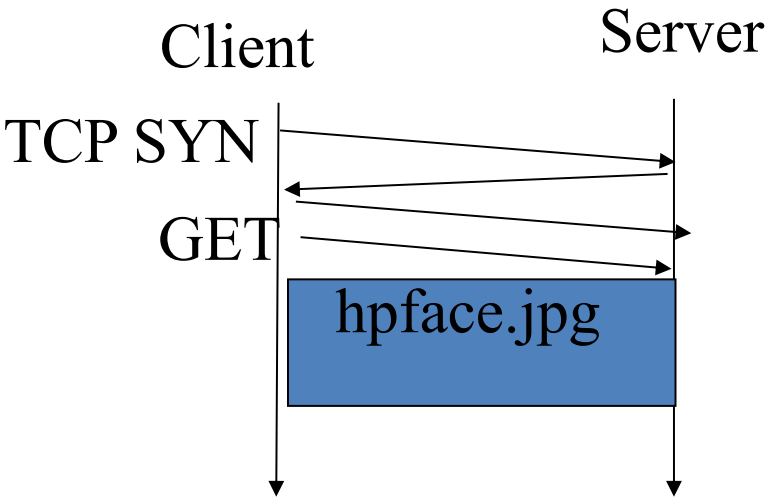
HTTP 1.1 builds on top of HTTP 1.0

- 3 Ps
  - Persistent
  - Parallelization
  - Pipeling

Persistent HTTP

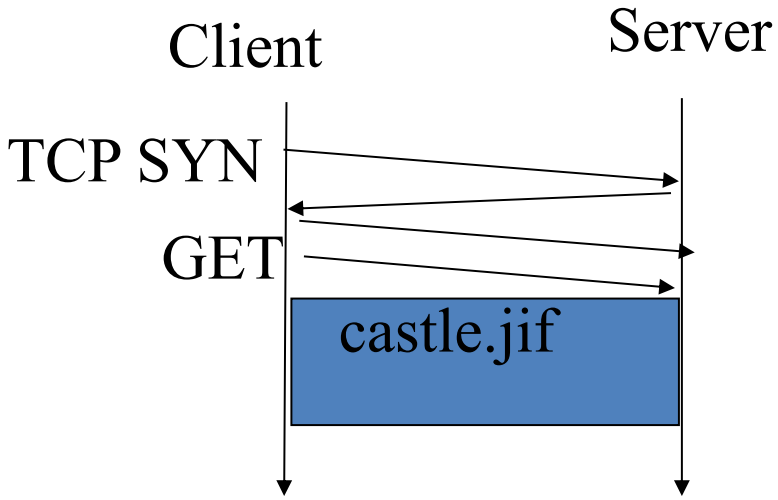


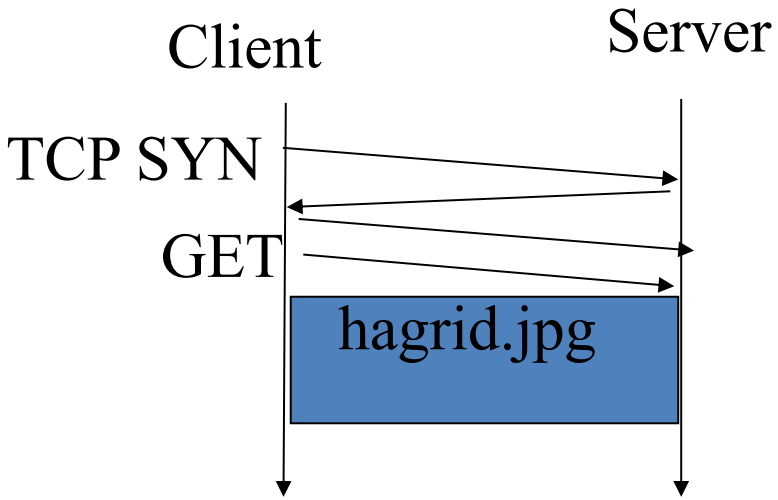
What is the time taken to transfer all the objects?



Parallel HTTP

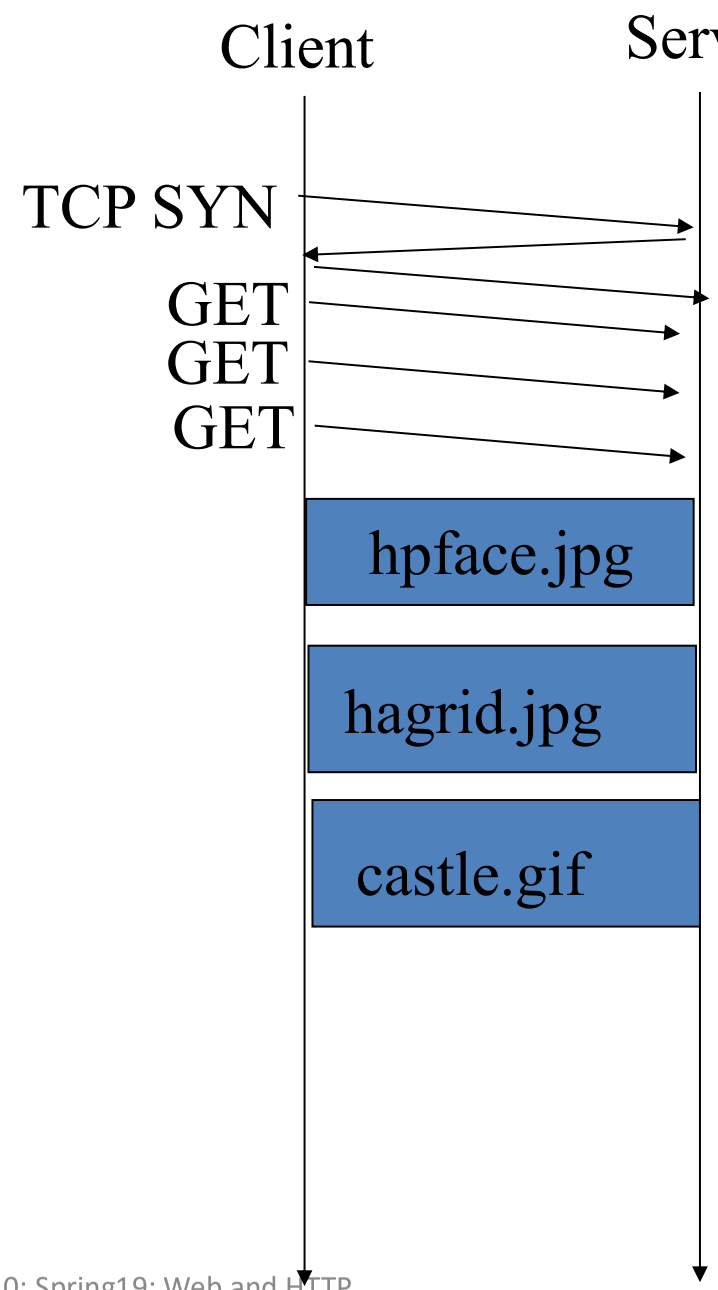
Browsers only allow
6 parallel connections per server*





What is total time taken?

Pipelined HTTP



What is total time taken?

In practice,

- HTTP 1.1 uses
  - Persistent
  - Parallelization
  - But not pipelining (because of head-of-line blocking)
- Newest HTTP protocol
  - HTTP/2 (will be covered as part of special topics)

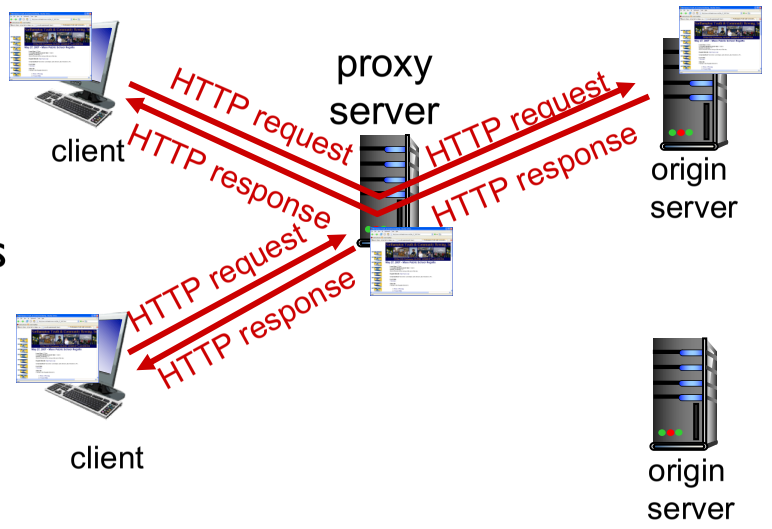
Head-of-line blocking

- In class.

Local and Web caches

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

- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- Internet dense with caches: enables "poor" content providers to effectively deliver content (so too does P2P file sharing)

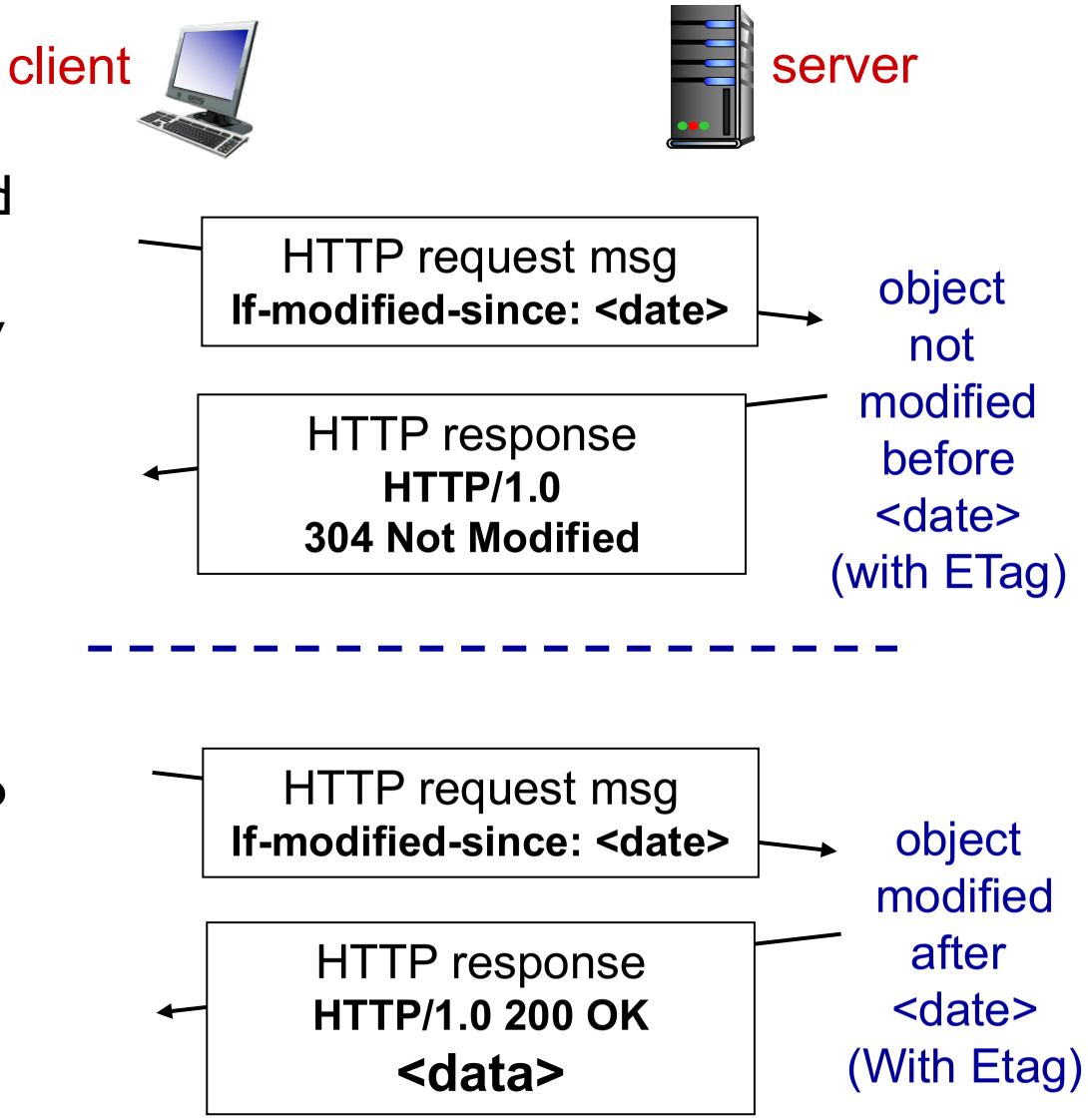
Conditional GET

- *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
```



Other cache primitives

- Cache Control
  - Max Age: Maximum age after which the resource is expired
  - No-Cache: Can cache a response but must first submit a validation
  - Public: Can be cached by any cache
  - Private: Can only be cached on the user device
  - No-Store: Cannot be cached.