BRAC UNIVERSITY

Inspiring Excellence

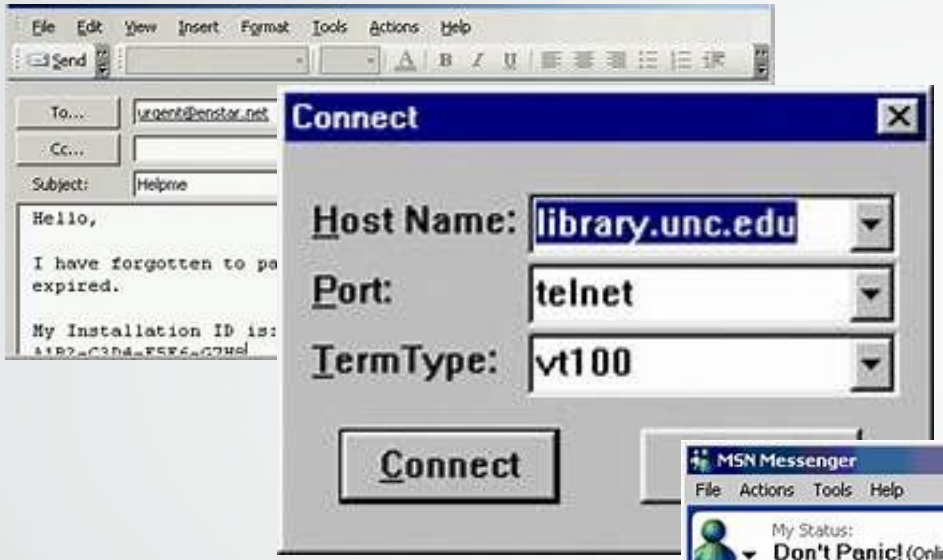
Application Layer (HTTP)

Lecture 2 | CSE421 – Computer Networks

Department of Computer Science and Engineering
School of Data & Science

Applications

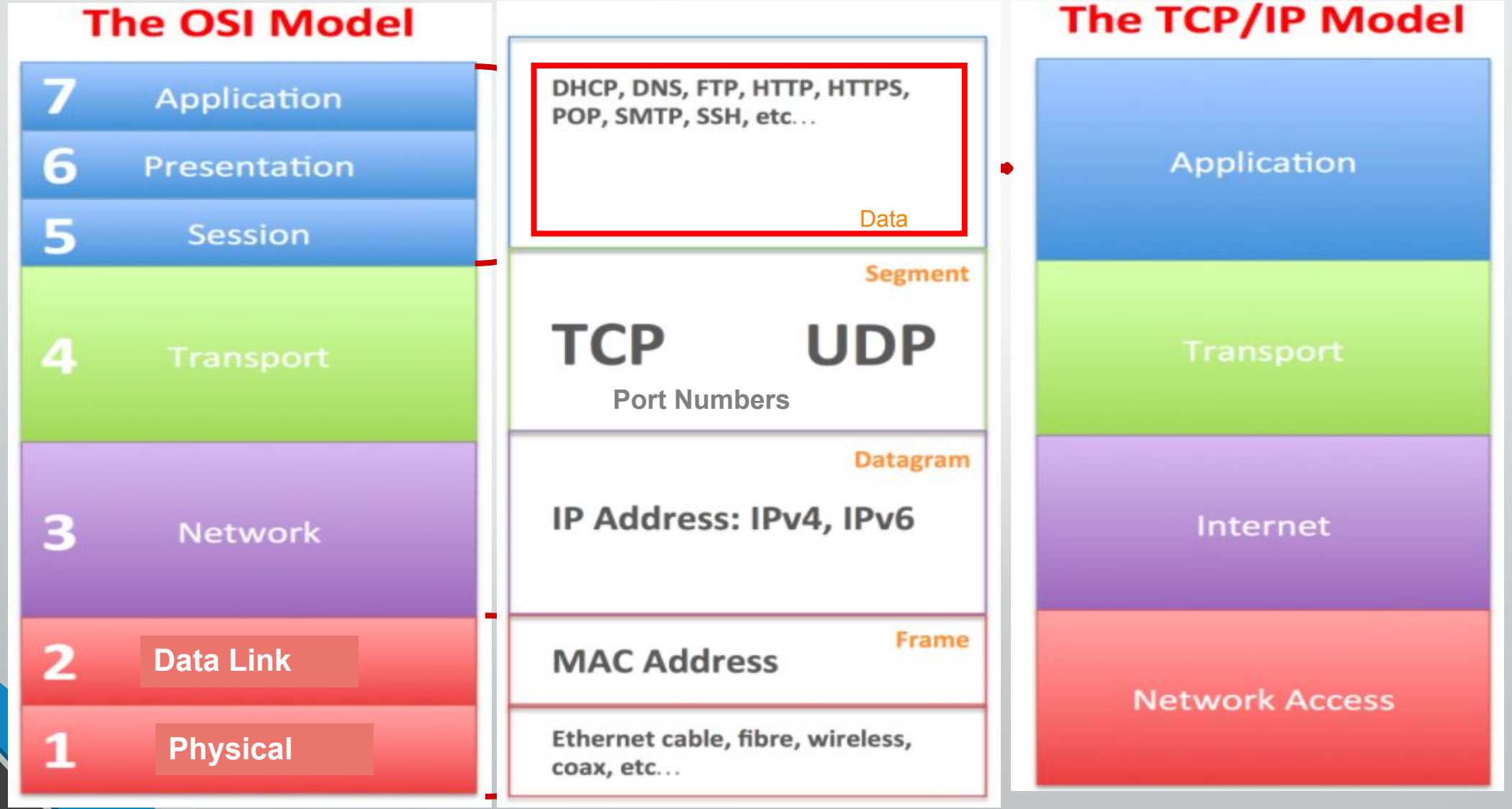
- 1970-1980
- 1990-2000
- 2000-2020



Objectives

- Principles of network applications
- Web and HTTP
- Electronic Mail (SMTP, POP3, IMAP)
- DNS
- P2P Applications
- Video streaming and content distribution networks

Network Models

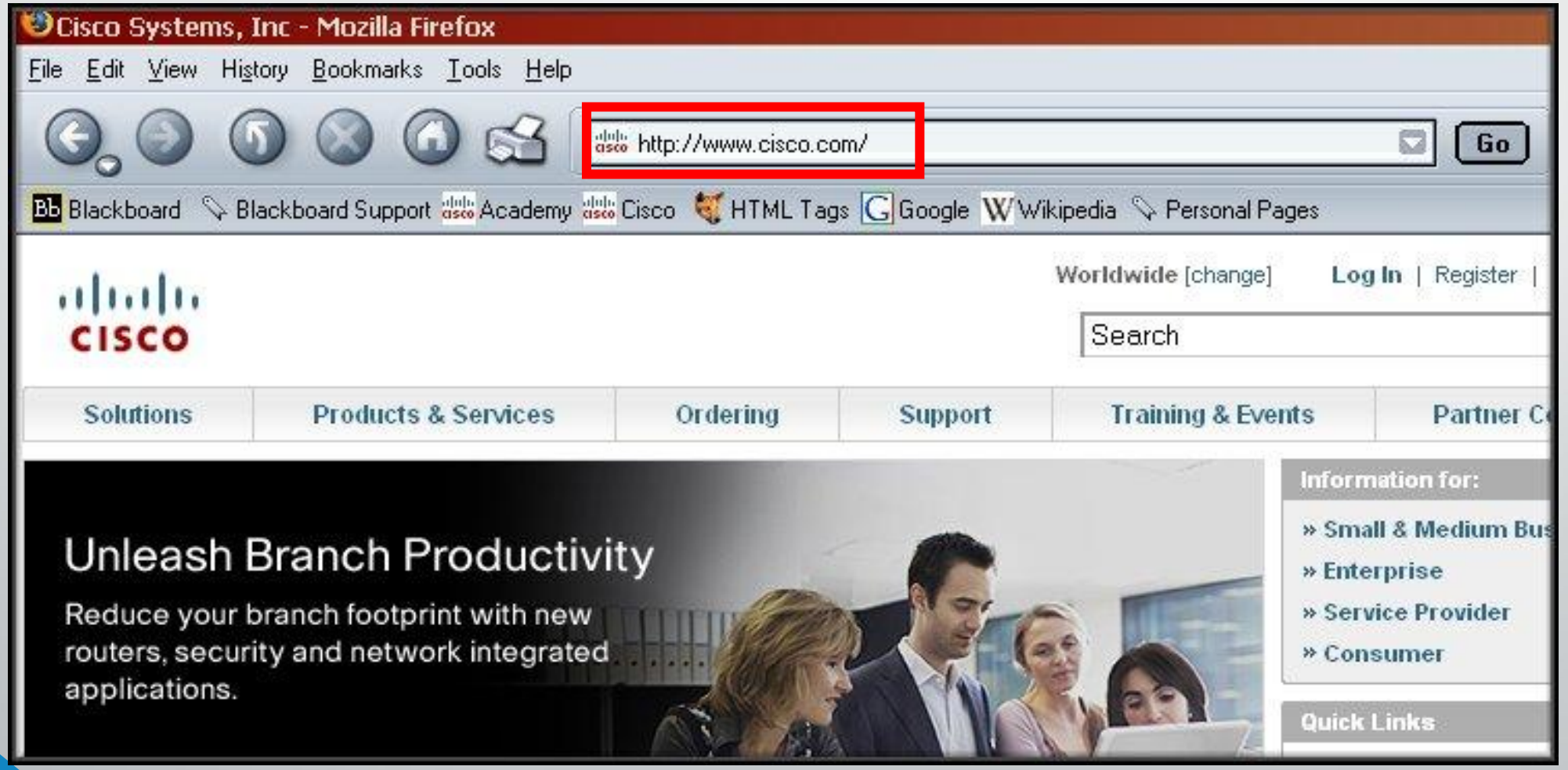


Application Layer

- Application Layer Protocols
  - Provide the rules and formats that govern how data is treated in the application layer.
- Application Software
  - The programs used to communicate over the network.

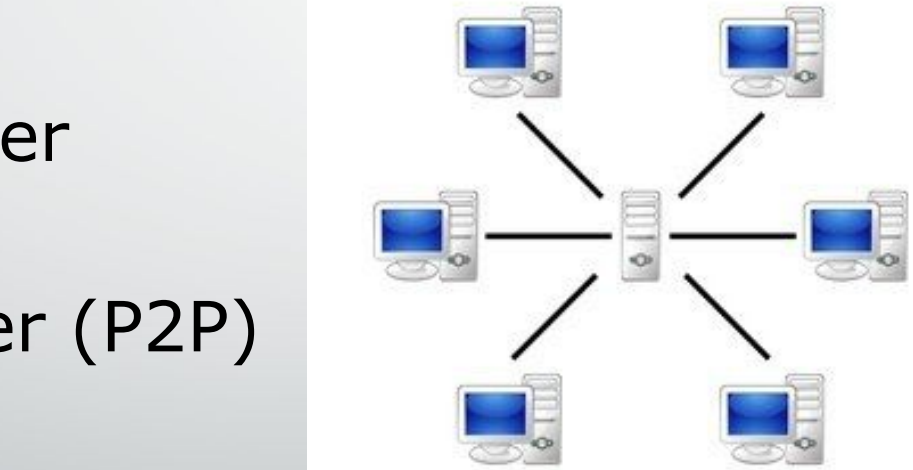
- For example:
  - When displaying a web page:
    - The Application Layer uses the HTTP(Hyper Text Transfer Protocol) Protocol.
    - The Application Software is your Web browser.

Application Layer

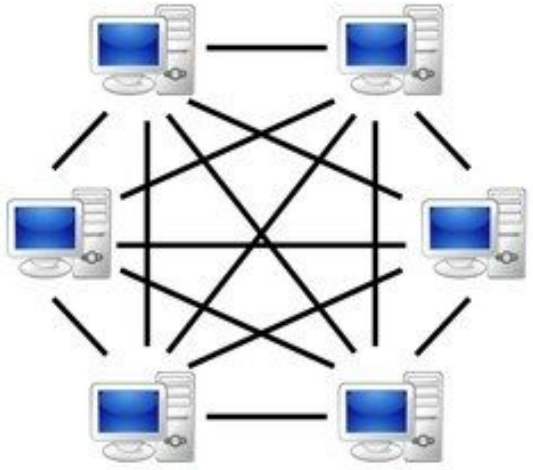


Application Layer

- When accessing information on a device, the data may not be physically stored on that device.
- If that is the case, a request must be made to the device where the data resides.
- **Two methods:**
  - Client/Server
  - Peer-to-Peer (P2P)



Client-Server

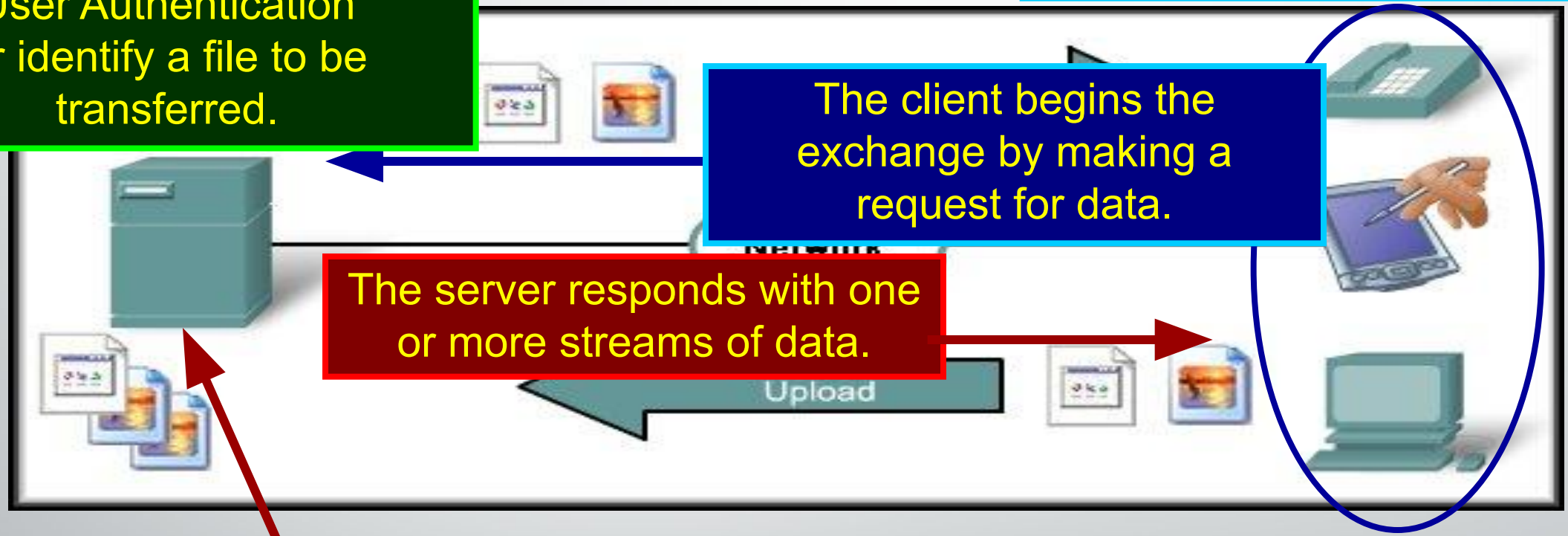


Peer-to-Peer(P2P)

Client/Server Model

May also require control information.
User Authentication
or identify a file to be transferred.

Clients – hardware, software combination

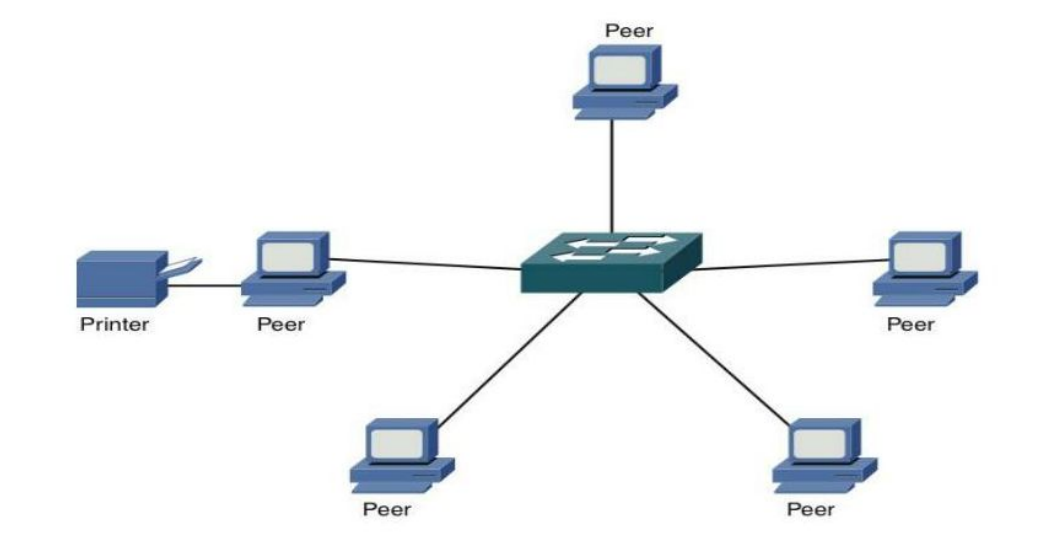


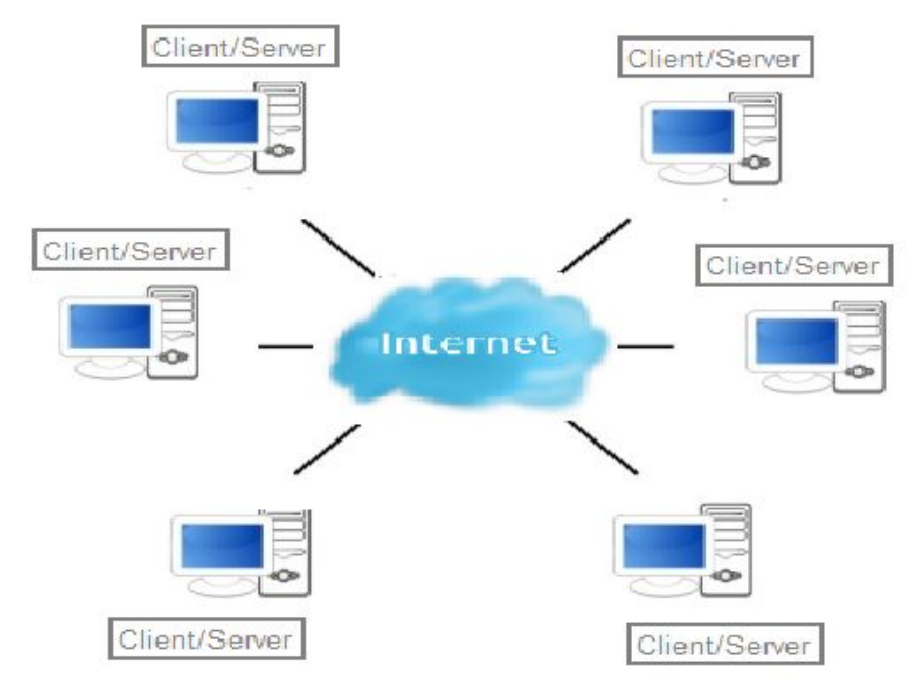
The client begins the exchange by making a request for data.

The server responds with one or more streams of data.

Resources are stored on the server.

Peer-to-Peer Model





- Two or more computers are connected via a network and can share resources (such as printers and files) *without having a dedicated server*.
- End devices (peers) can function as either a *server or client* depending upon the required service.

Web and HTTP

Part 2

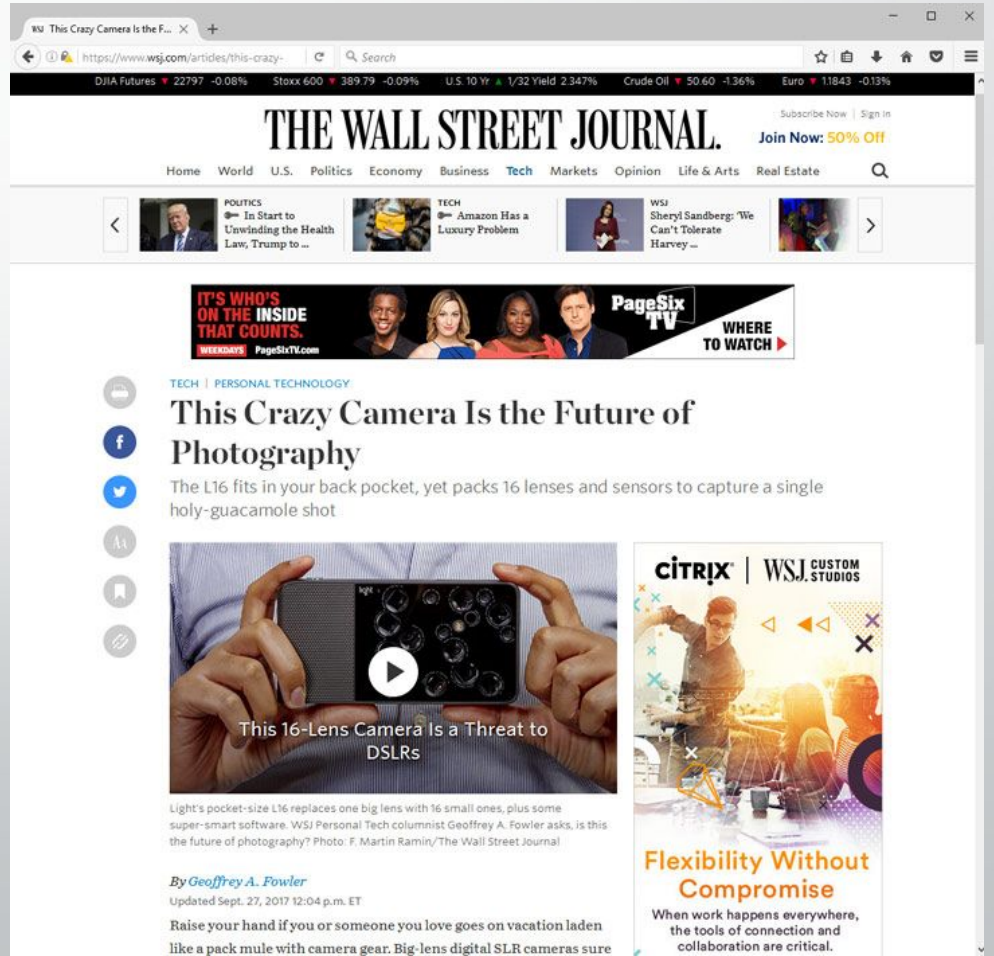
Objectives -Part 2

- WWW – The Web
- HTTP
- HTTP Messages and Methods

WWW- The Web

- A **web page** or **webpage** is a document that is viewed in an Internet browser.
- It may contain text, graphics, and hyperlinks to other web pages and files.

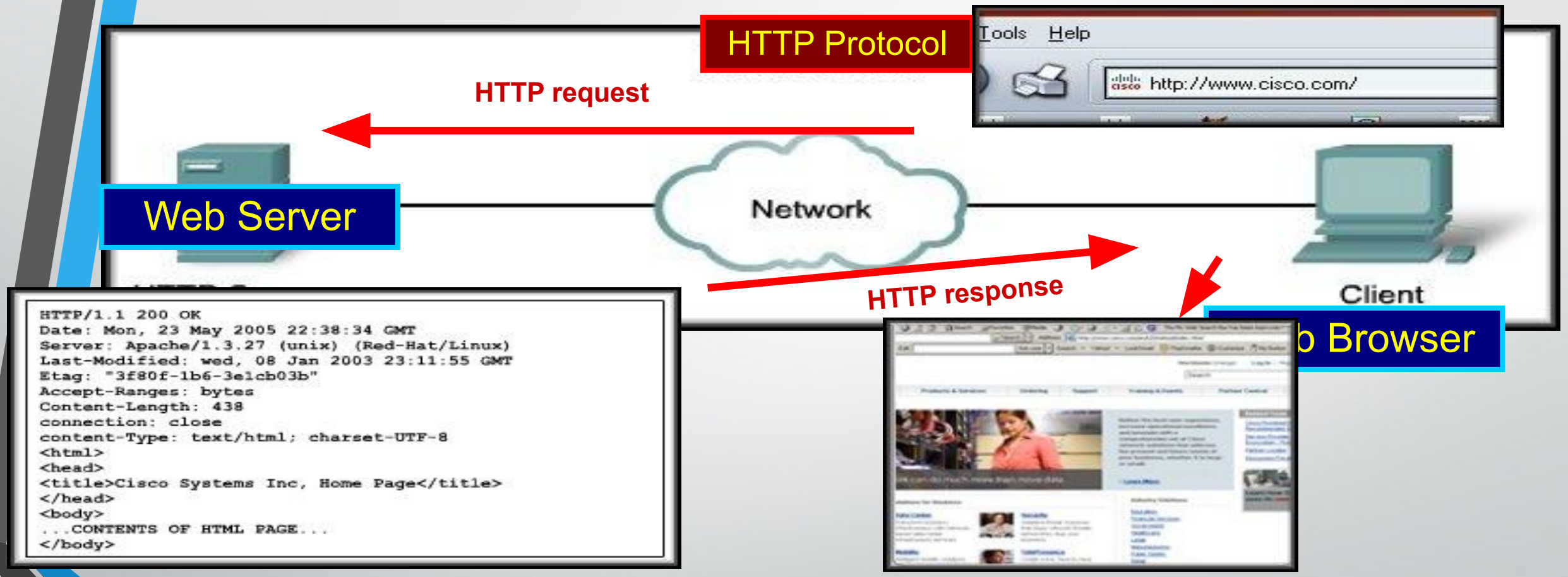
- ***A web page*** consists of *objects*
- Web page consists of *base HTML-file* which includes *several referenced objects*
- Objects can be HTML file, JPEG image, Java applet, audio file…



WWW- The Web

- A web page can be accessed by entering a URL address into a web browser's address bar.

HTTP



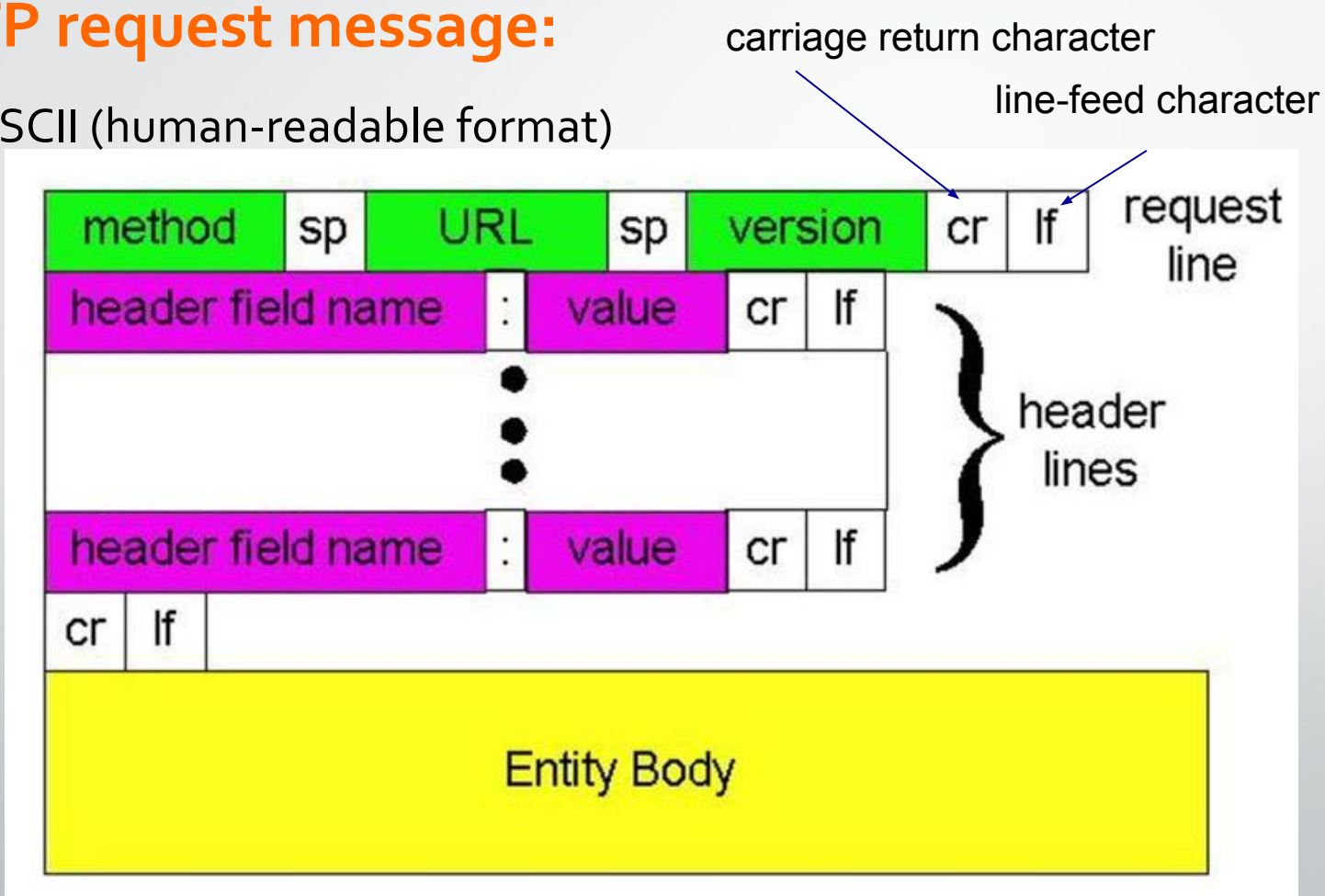
- Web browsers are the client applications used to interpret the HTTP application protocol received from a web server.

HTTP messages

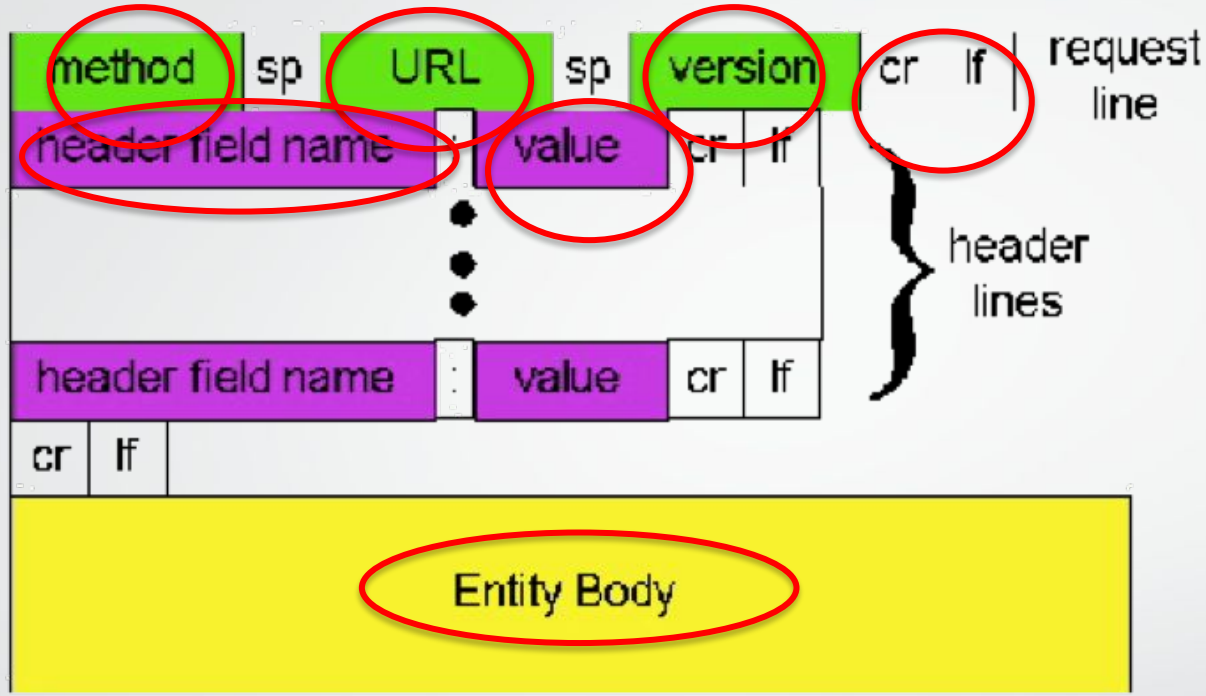
- Two types of HTTP messages:
- *HTTP Request* message
- HTTP *Response message*

HTTP request message format

- **HTTP request message:**
  - ASCII (human-readable format)



HTTP request message: Example



request line (GET, POST, HEAD,PUT,DELETE commands)

header lines

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

```
GET /somedir/index.html HTTP/1.1\r\n
Host: www.someschool.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,image/gif, image jpeg\r\n
Accept-Language: fr \r\n
Connection: close\r\n

\r\n
```

HTTP Method types

HTTP/1.0:

- GET
  - Primarily gets information only
  - Retrieve data
- POST
  - Creating new data
- HEAD
  - Also retrieve but asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT/PATCH
  - Update data
  - Replaces existing objects
- DELETE
  - Deletes data

Uploading form input

POST method:

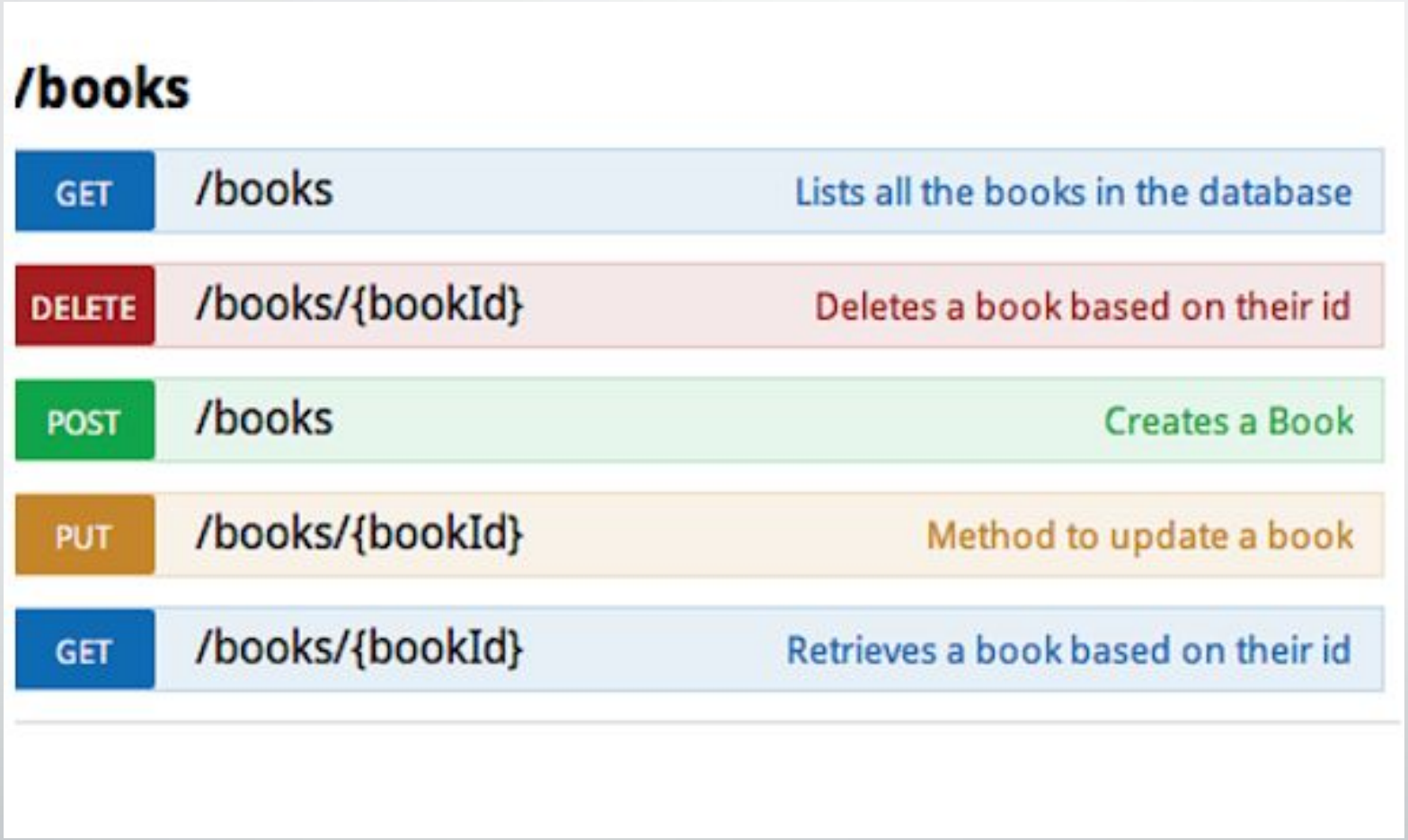
- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

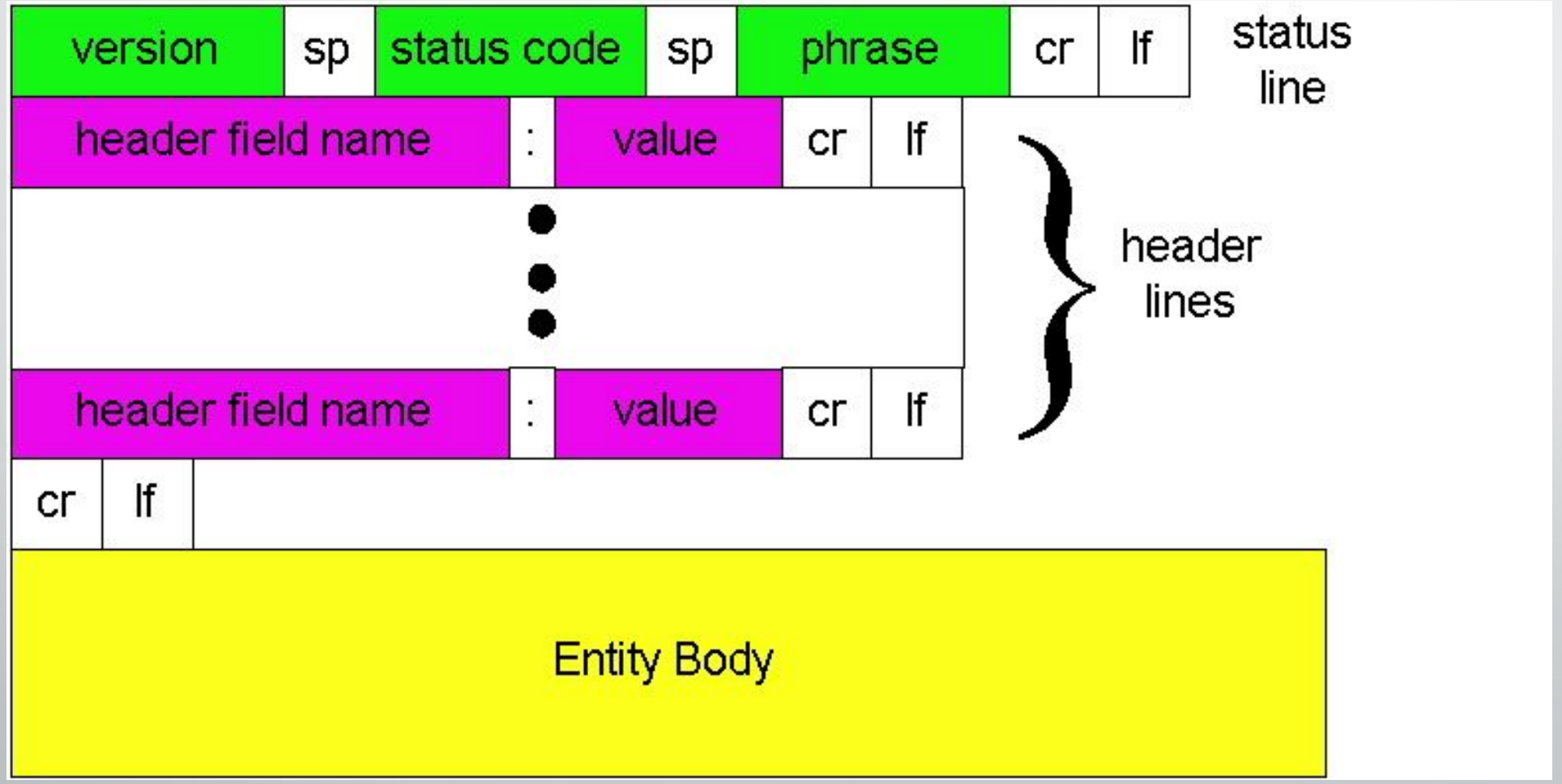
```
www.somesite.com/animalsearch?monkeys&banana
```

HTTP Methods

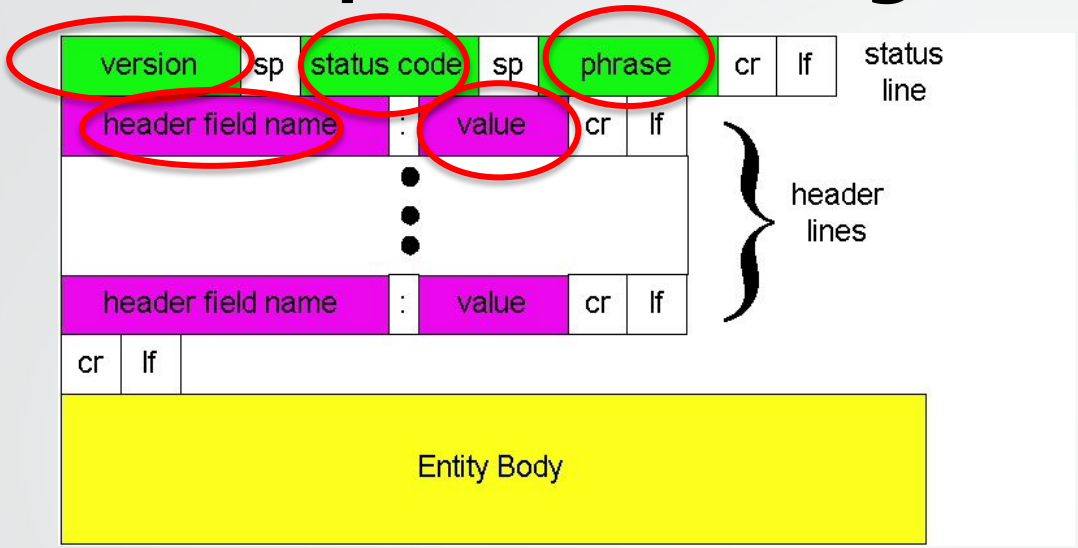


Source :https://javarevisited.blogspot.com/2016/04/what-is-purpose-of-http-request-types-in-RESTful-web-service.html

HTTP response message: General Format



HTTP response message : Example



Status line (protocol status code status phrase)

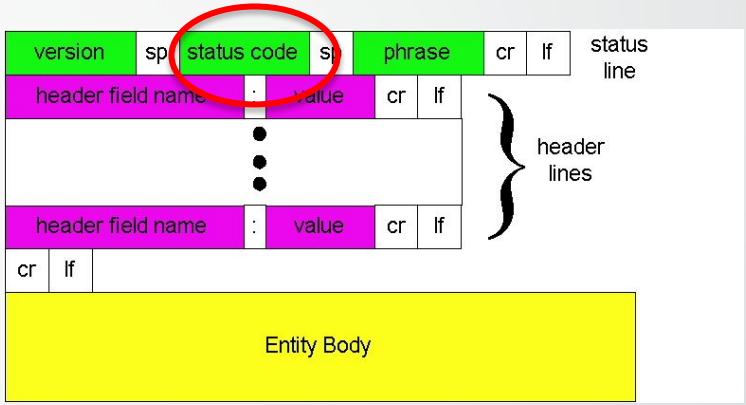
Header lines

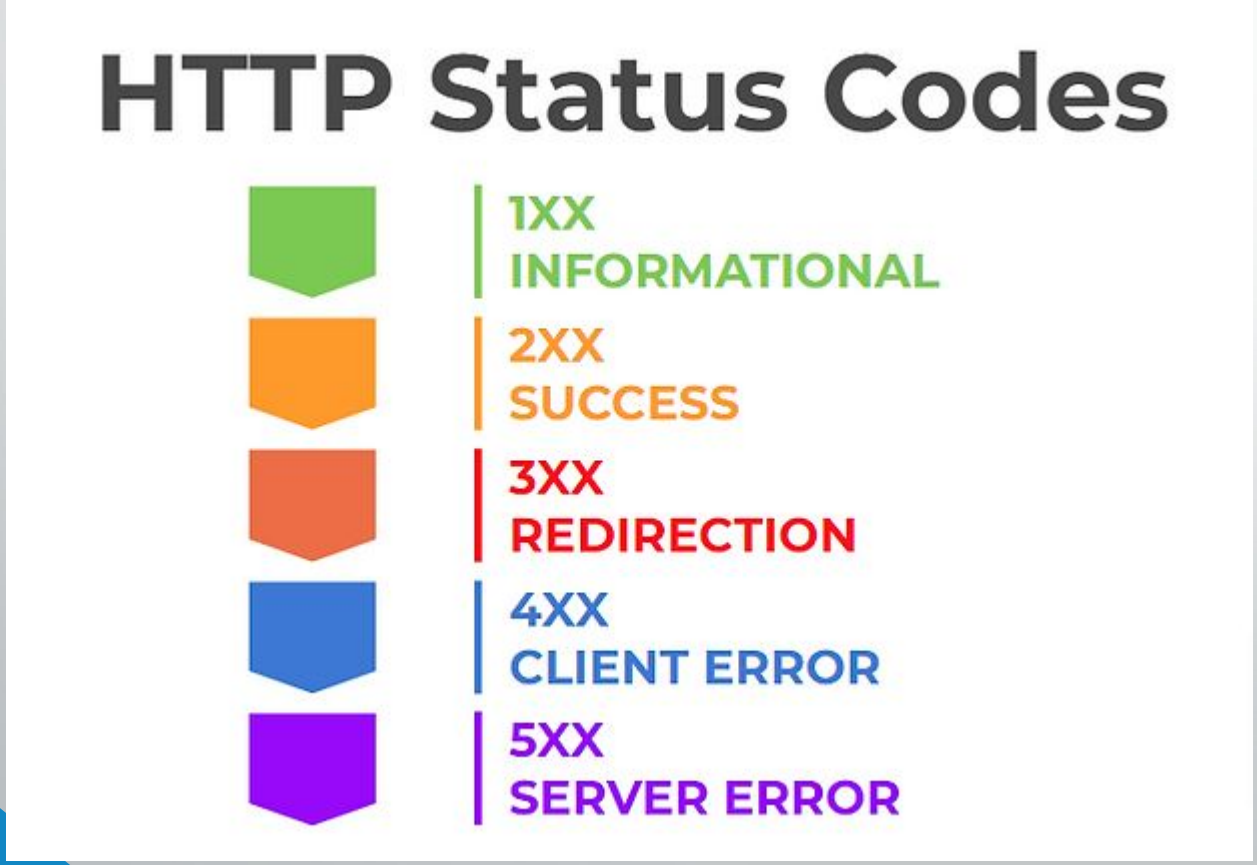
data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 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: close\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

HTTP response status codes

- Status code appears in 1st line in server-to-client response message.
- HTTP Status codes:





HTTP response status codes

- Some sample codes:

100 Continue

- everything ok so far, client should continue with the request

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

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

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Objectives -Part 3

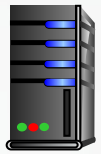
- HTTP Connections
  - Non Persistent
  - Persistent

HTTP Connections

Suppose user enters URL:

`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

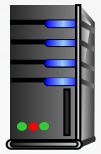
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

HTTP Connections (cont.)



5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

4. HTTP server closes TCP connection.

6. Steps 1-5 repeated for each of 10 jpeg objects

time

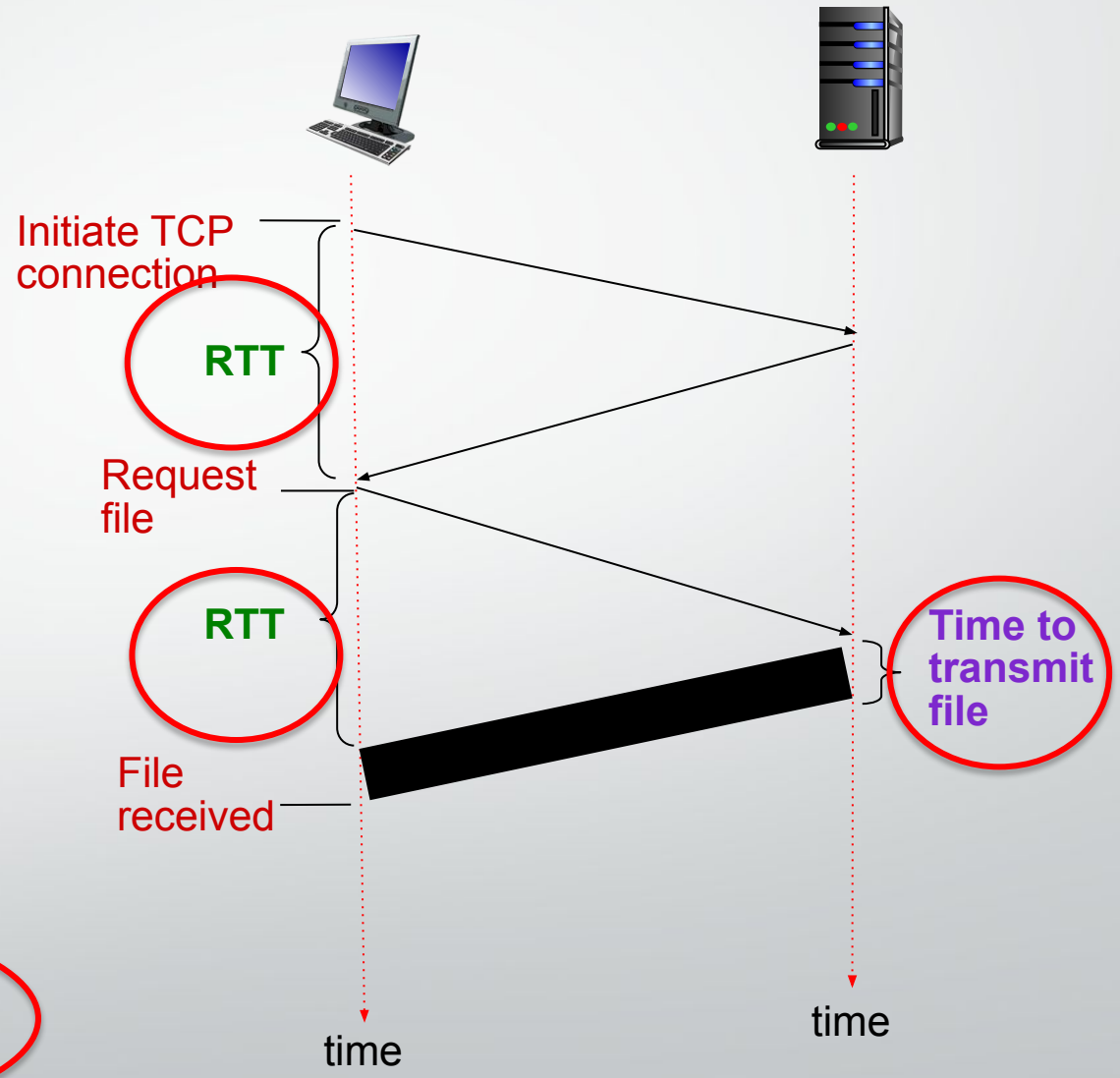
Non-persistent HTTP

Non-persistent HTTP: response time

RTT (Round Trip Time): time for a small 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
- File transmission time
- Non-persistent HTTP response time = 2*RTT+ file transmission time

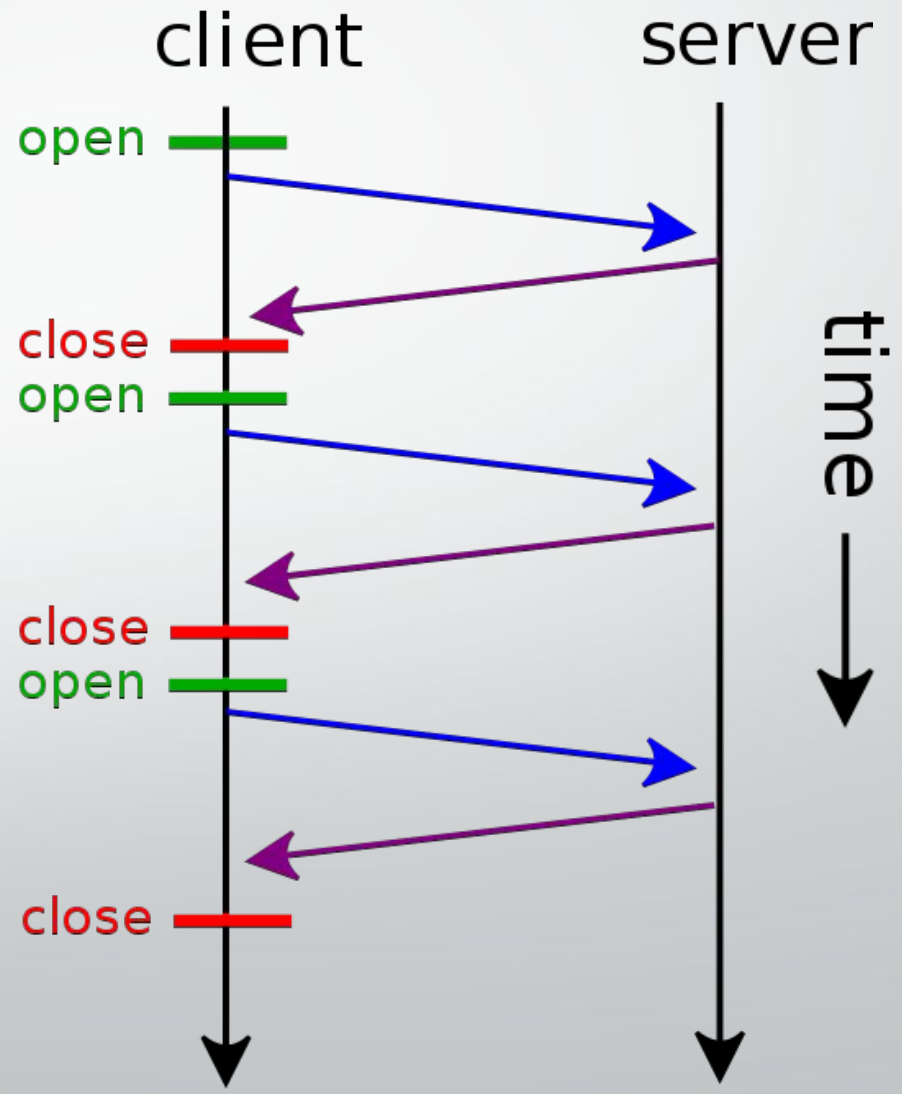


HTTP connections

Non-persistent HTTP

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

open ---TCP Connection Request
close --- TCP Termination Request



HTTP Transmission Calculation

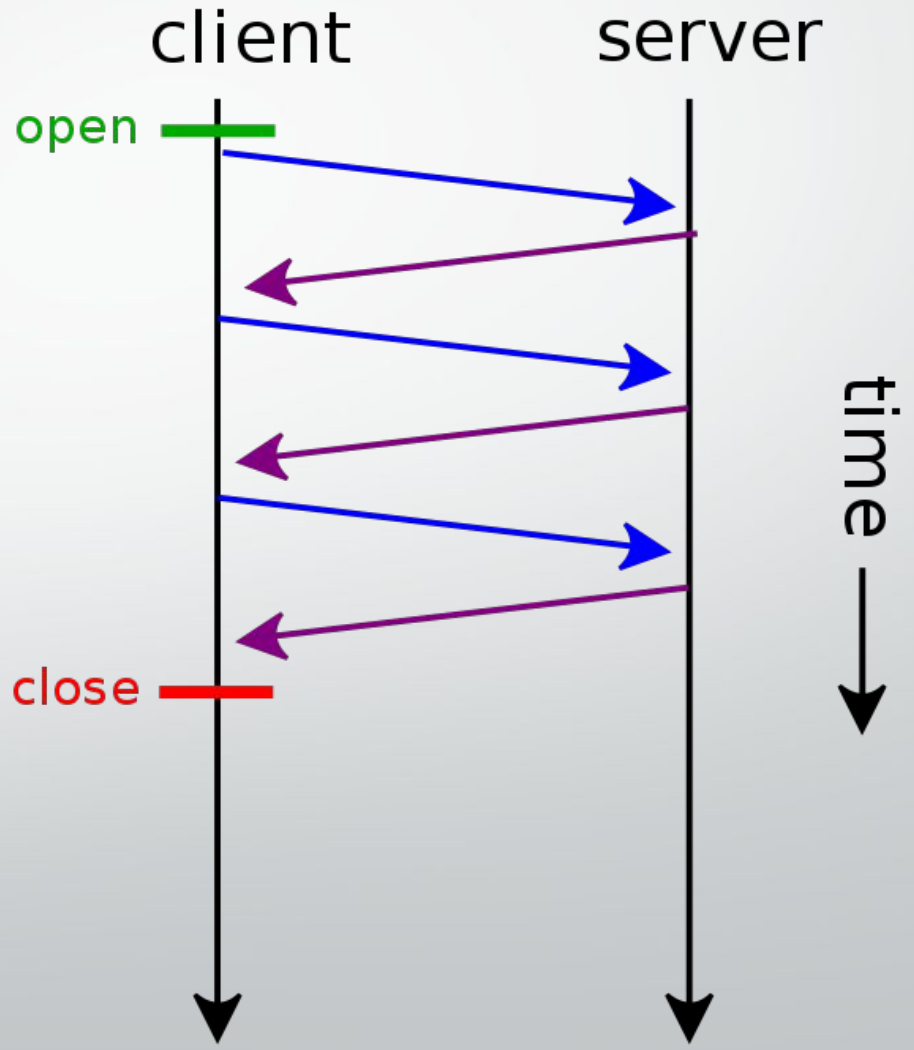
- You wish to access the web page "www.games.com", so you write the URL in your web browser.
  The Web page associated with the link contains exactly **10 object/s** in total. Each object size is **10 MB**.
- Let **25 ms** denote the time required to send a TCP request from PC to server.
- Assuming **100 Mbps** the transmission speed of the web server
- **Non-persistent** HTTP is being used.
- Please calculate the total time in ms that will take the web browser to load your page.

HTTP connections

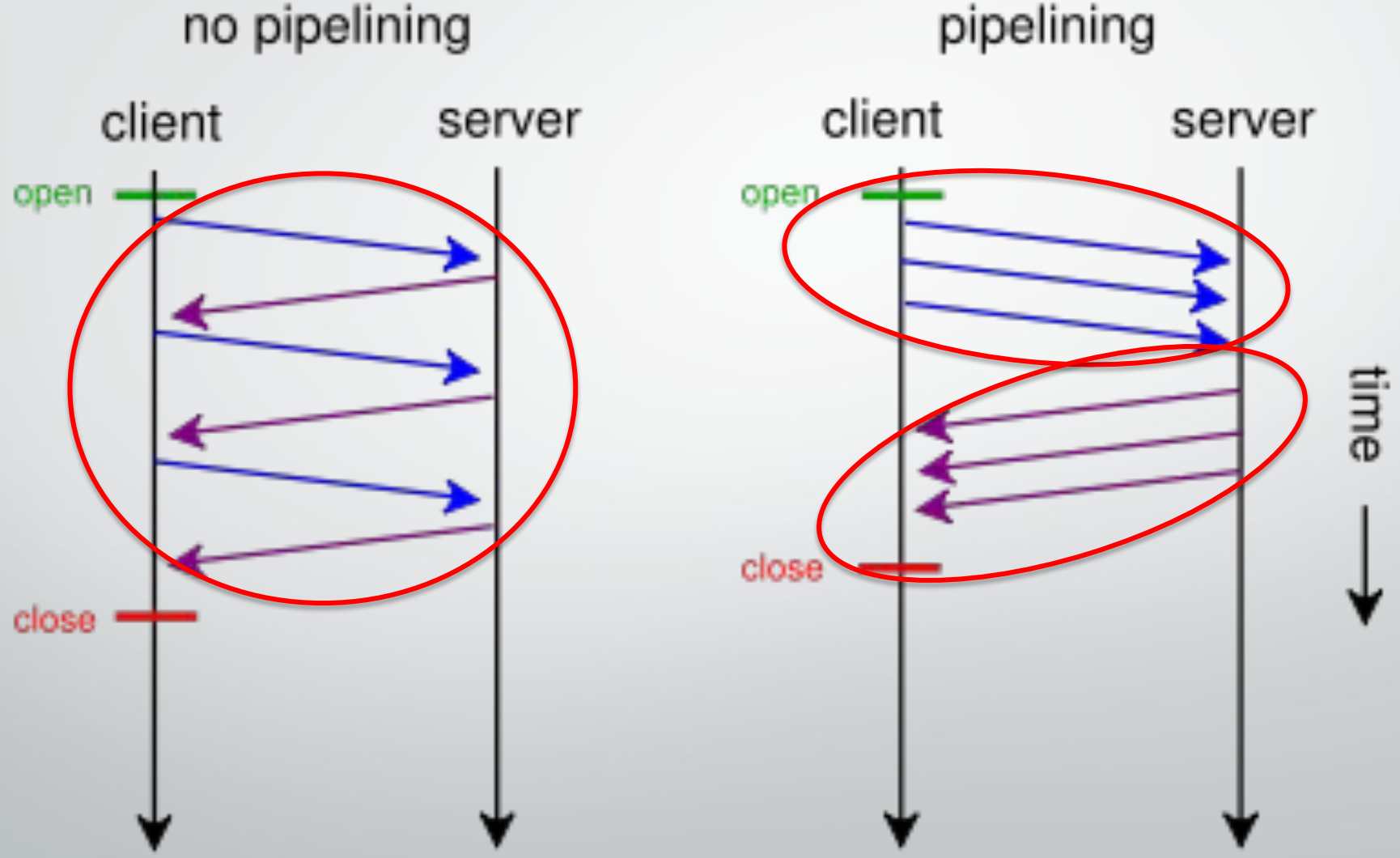
Persistent HTTP

- Multiple objects can be sent over single TCP connection between client, server

open ---–TCP Connection Request
close ---– TCP Termination Request



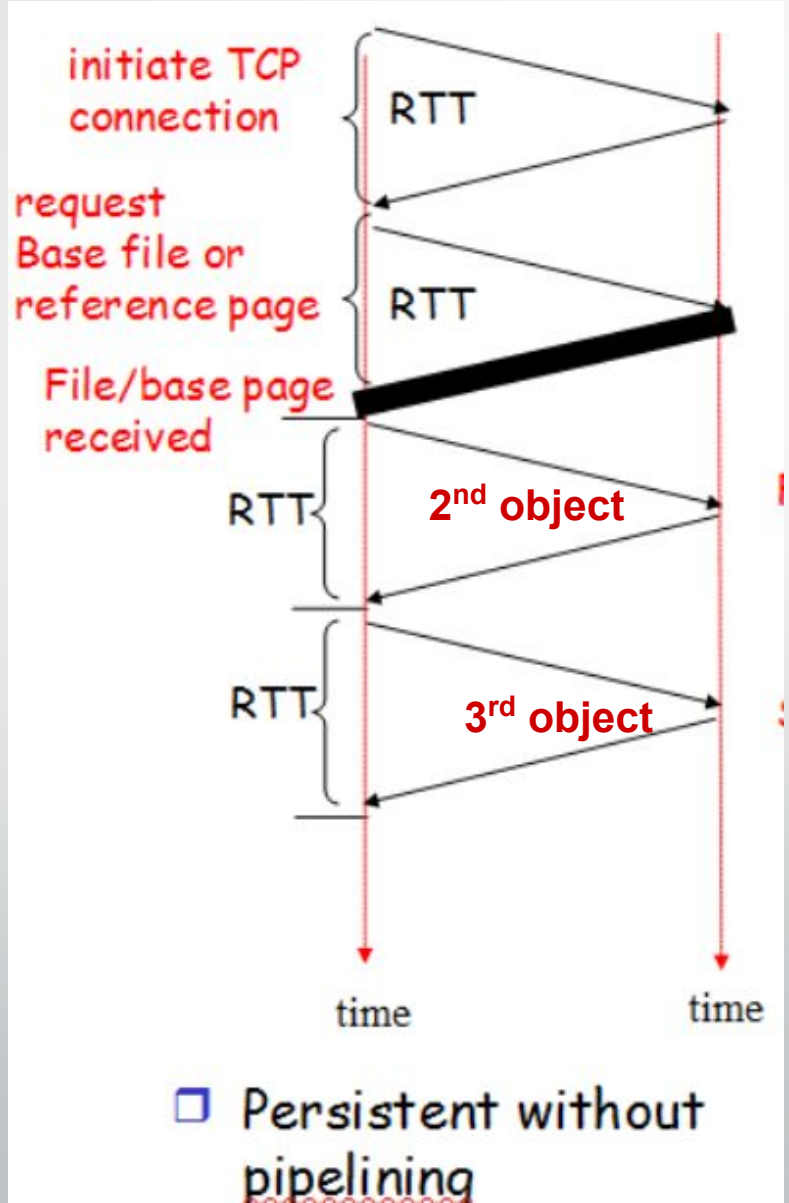
HTTP connections



Persistent HTTP

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



Persistent HTTP:

- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects

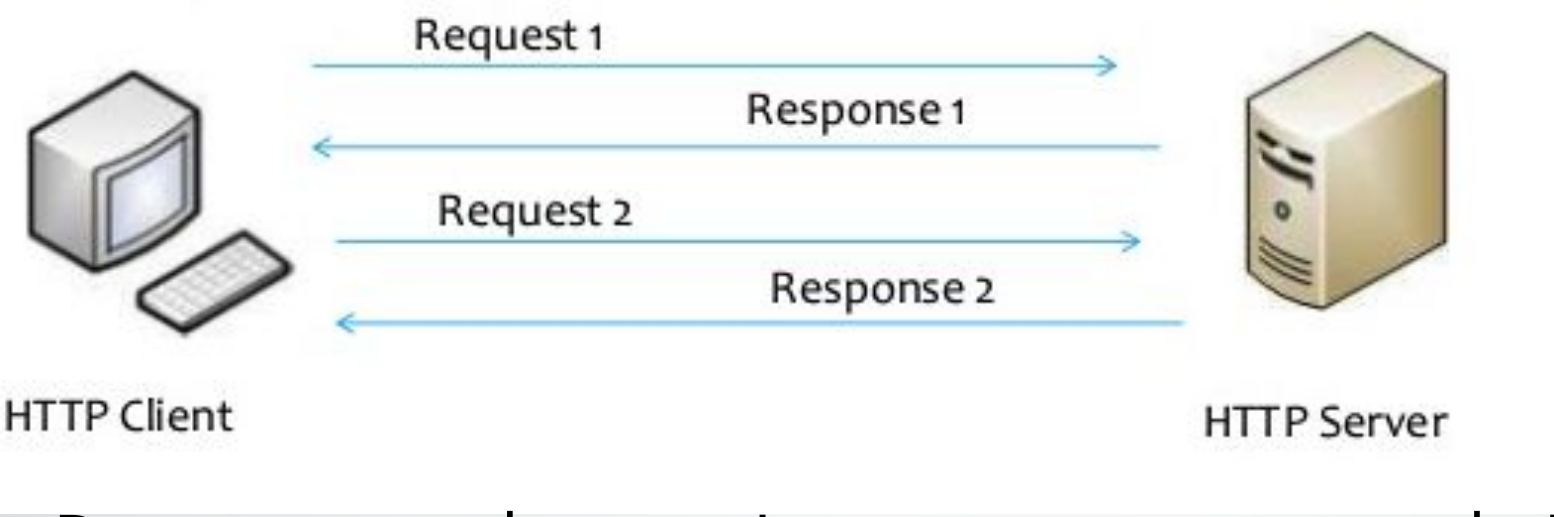
COOKIES

Part 4

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

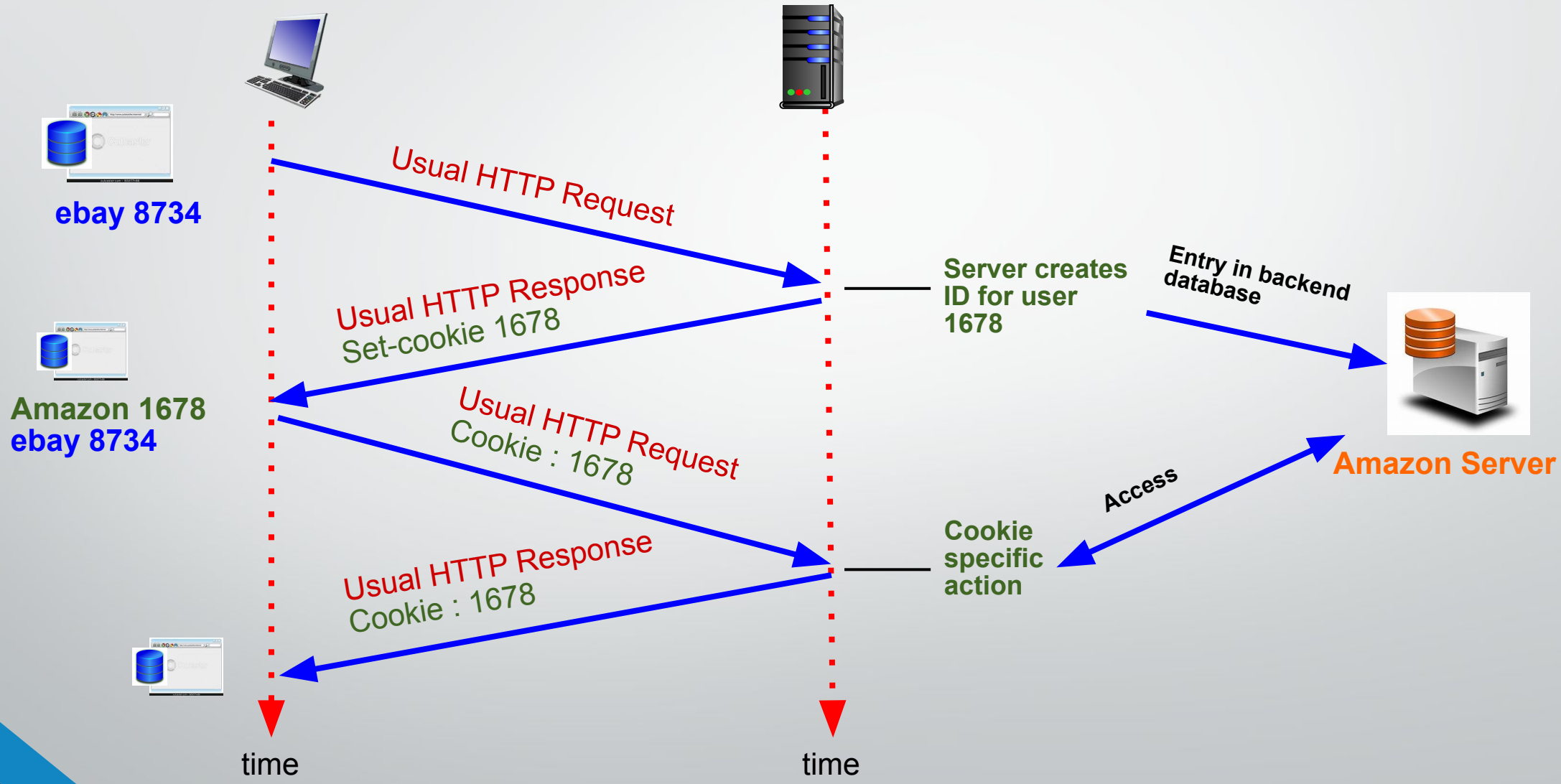
Four components:

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

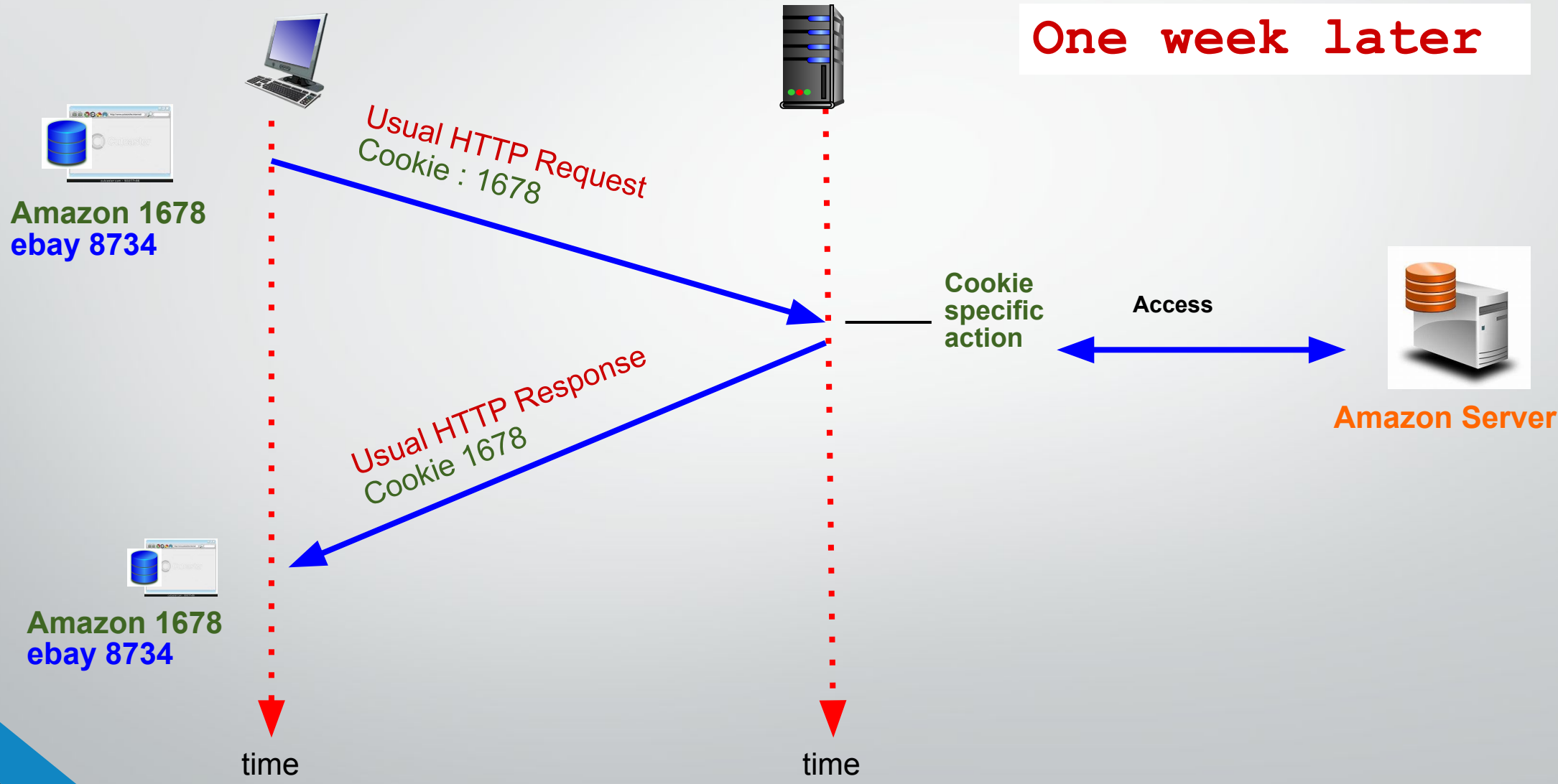
Example:

- 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)

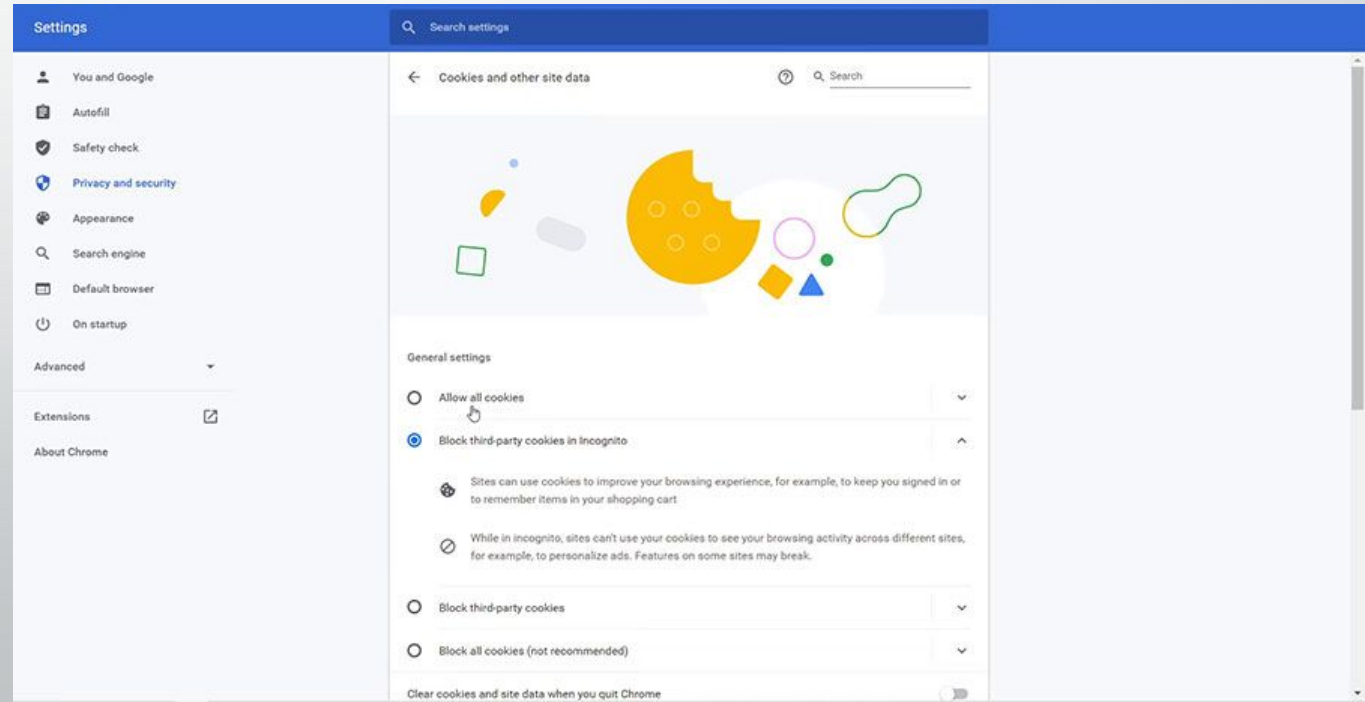
How to keep "state":

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

aside

Cookies and privacy

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



Web Cache or Proxy Server

Part 5

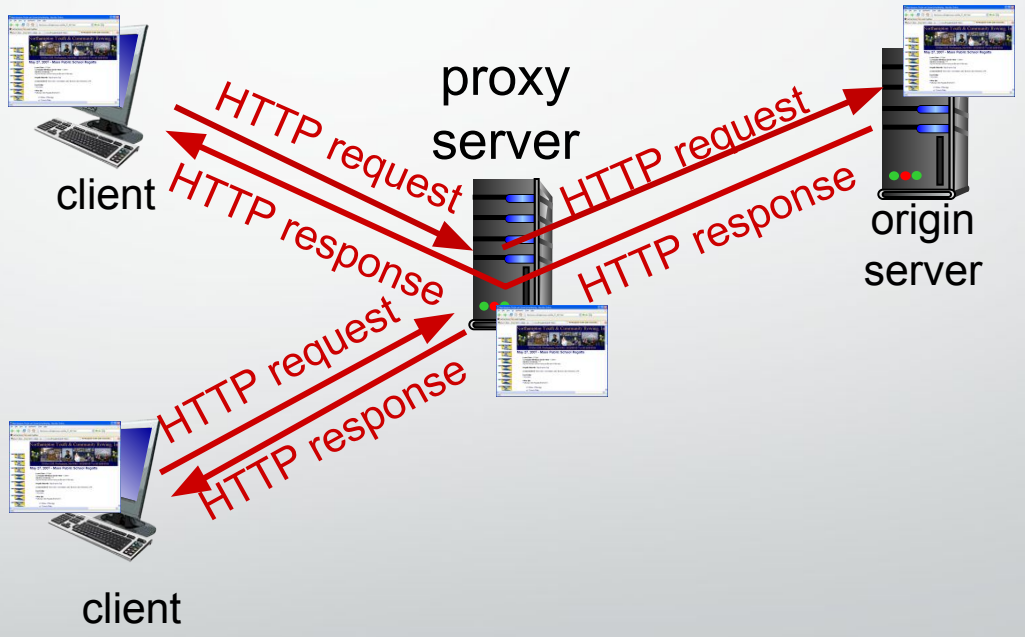
Objectives - Part 5

- 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

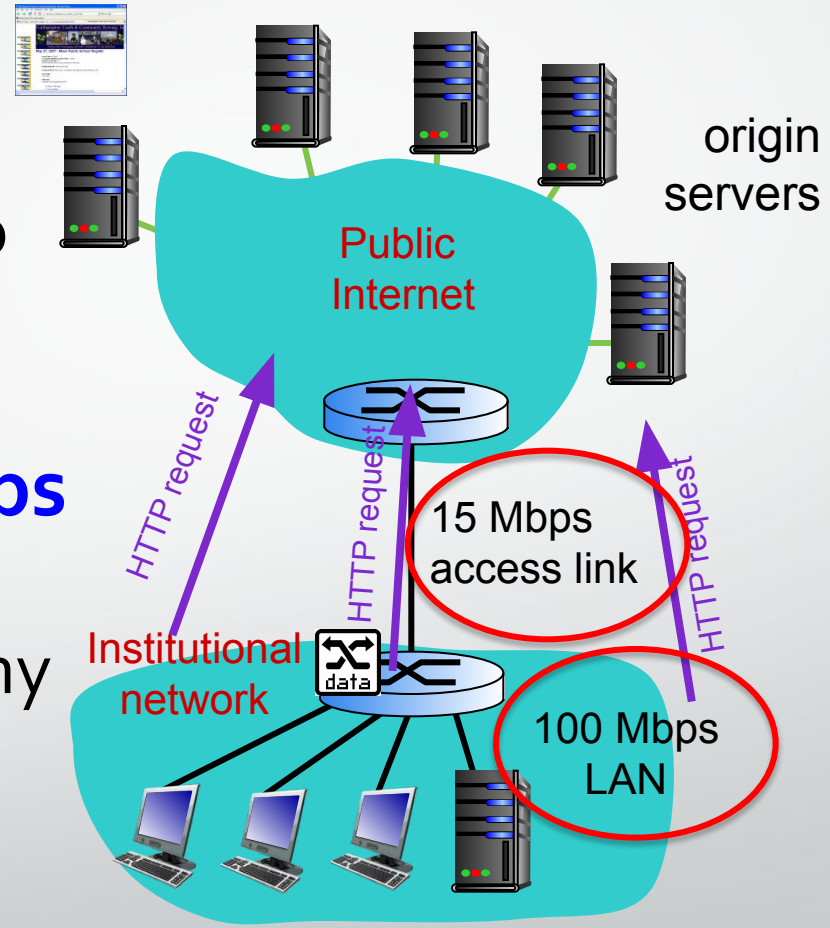
Advantages of Web Caching

- 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)

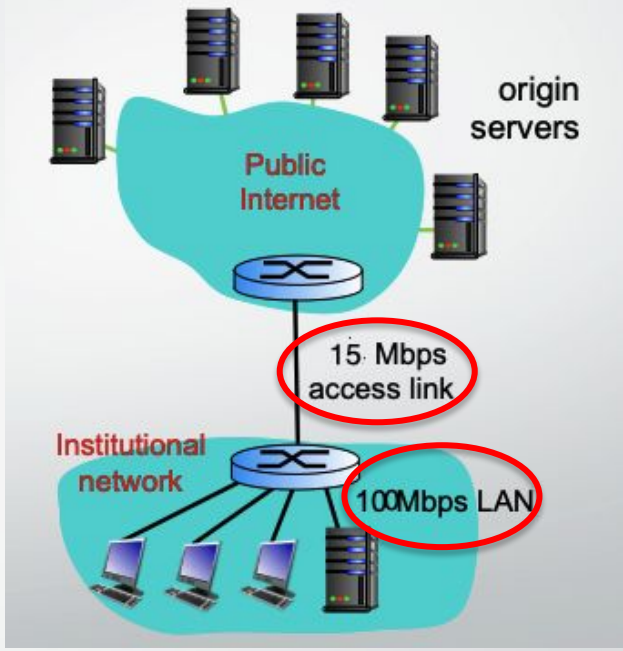
Caching example:

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

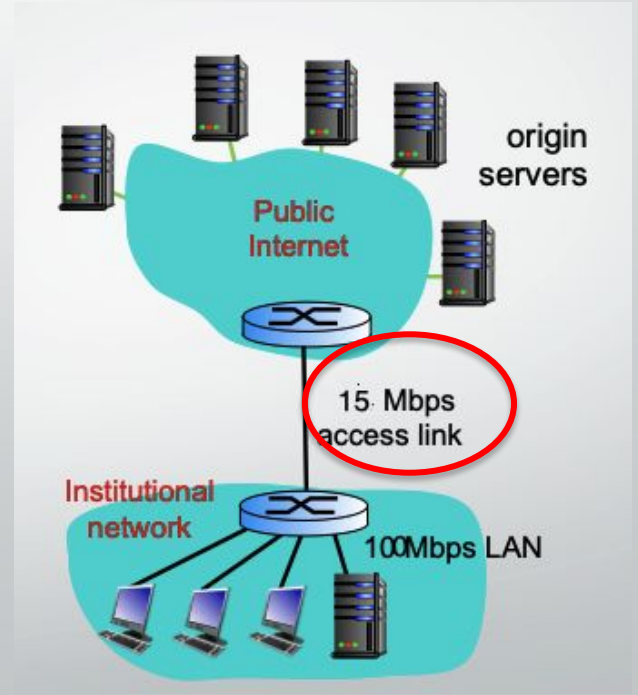
Consequences:

- 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.

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

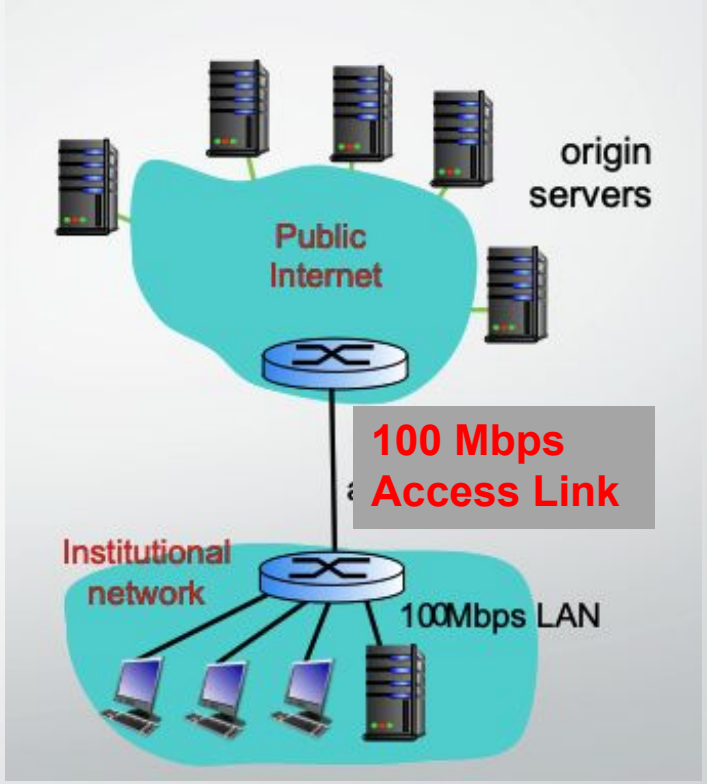
Caching example: Solution 1

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 + ~~secs~~ msecs

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

Cost: increased access link speed (not cheap!)

Caching example: Solution 2

Assumptions:

- Average hit rate: 40%

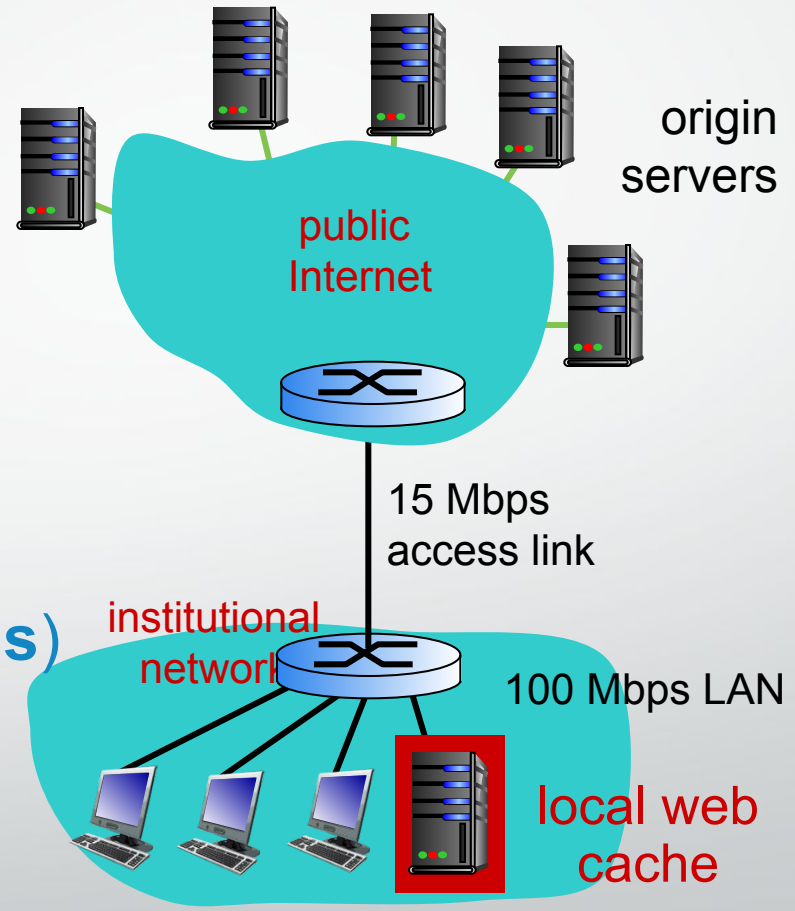
Consequences:

- **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.

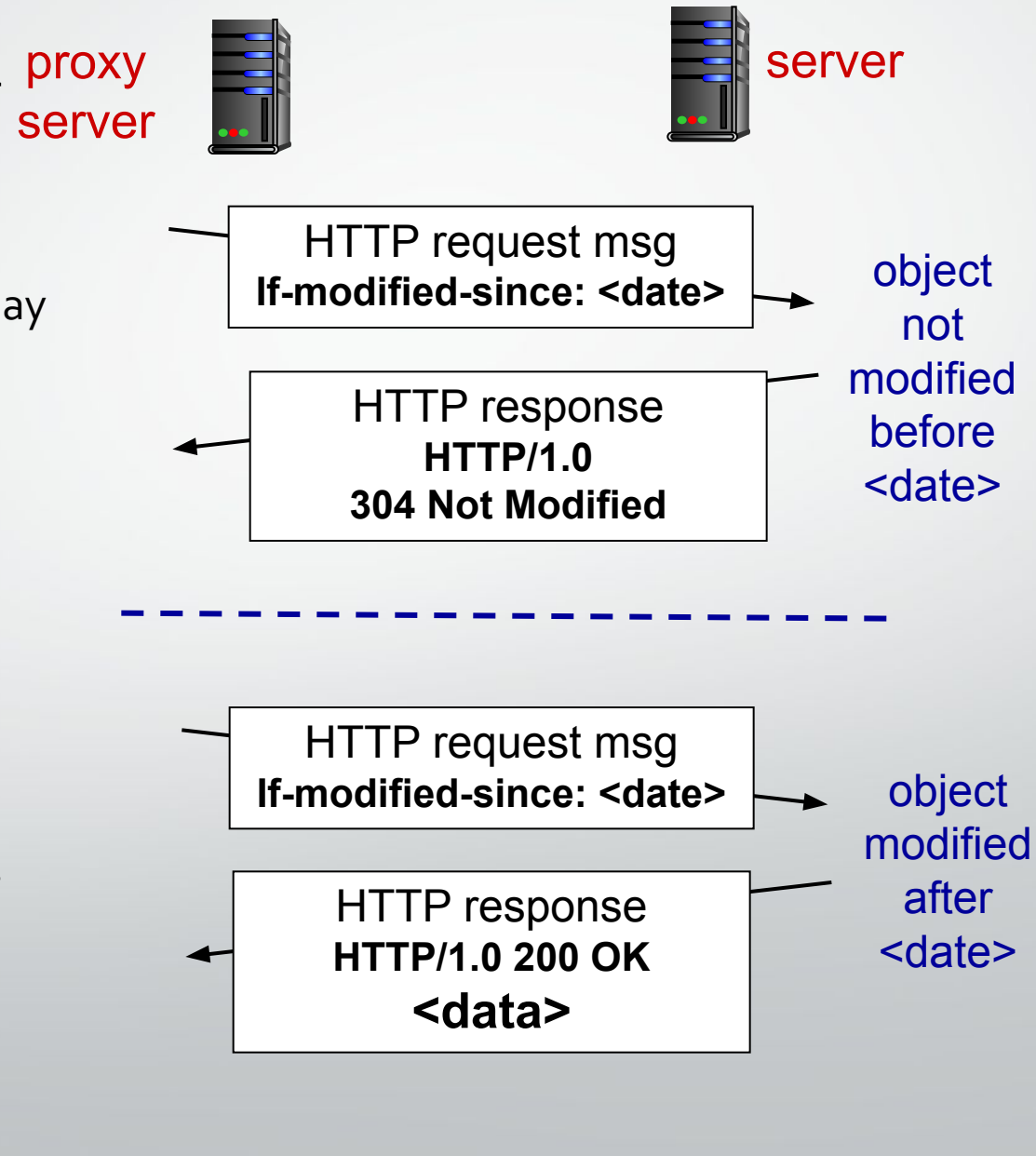
Conditional GET

- ***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
```



Different versions of HTTP

Part 6

HTTP/2

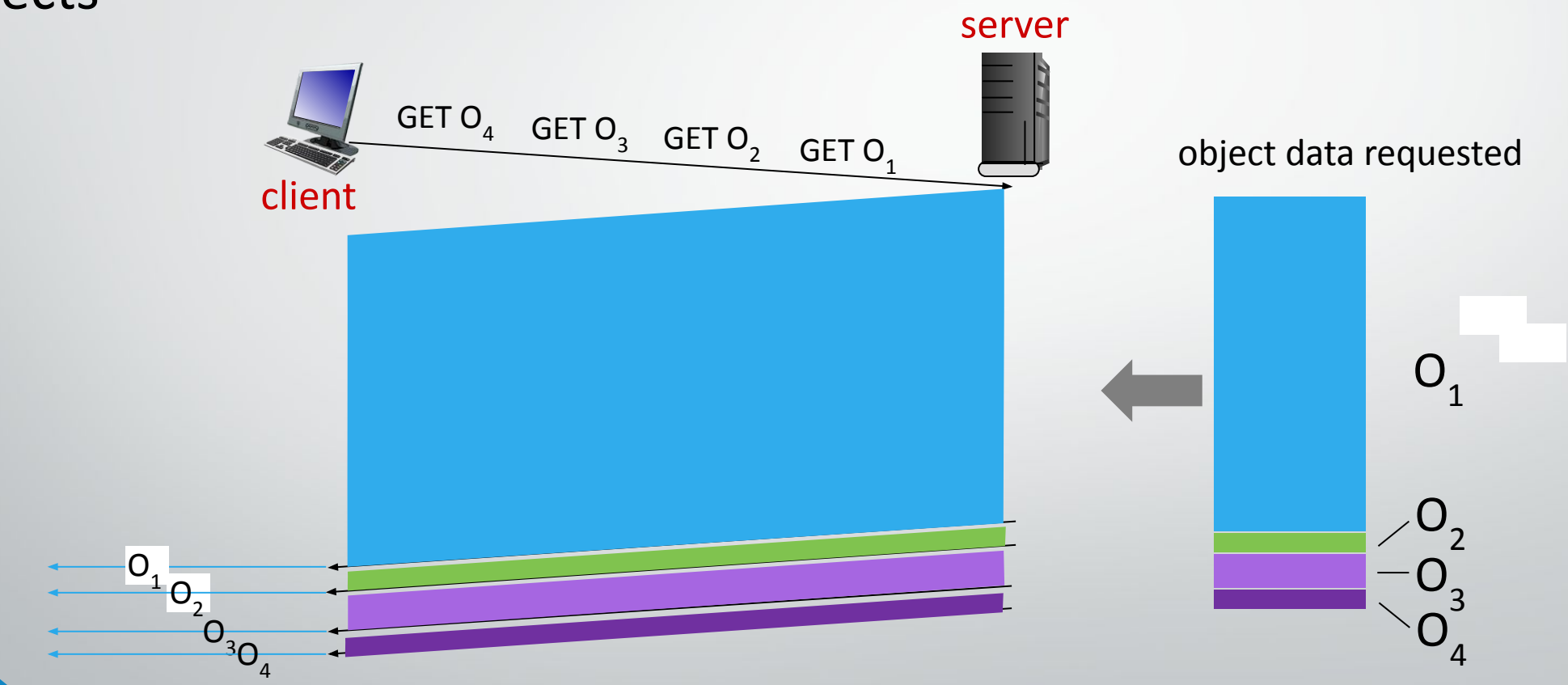
Key goal: decreased delay in multi-object HTTP requests

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)

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

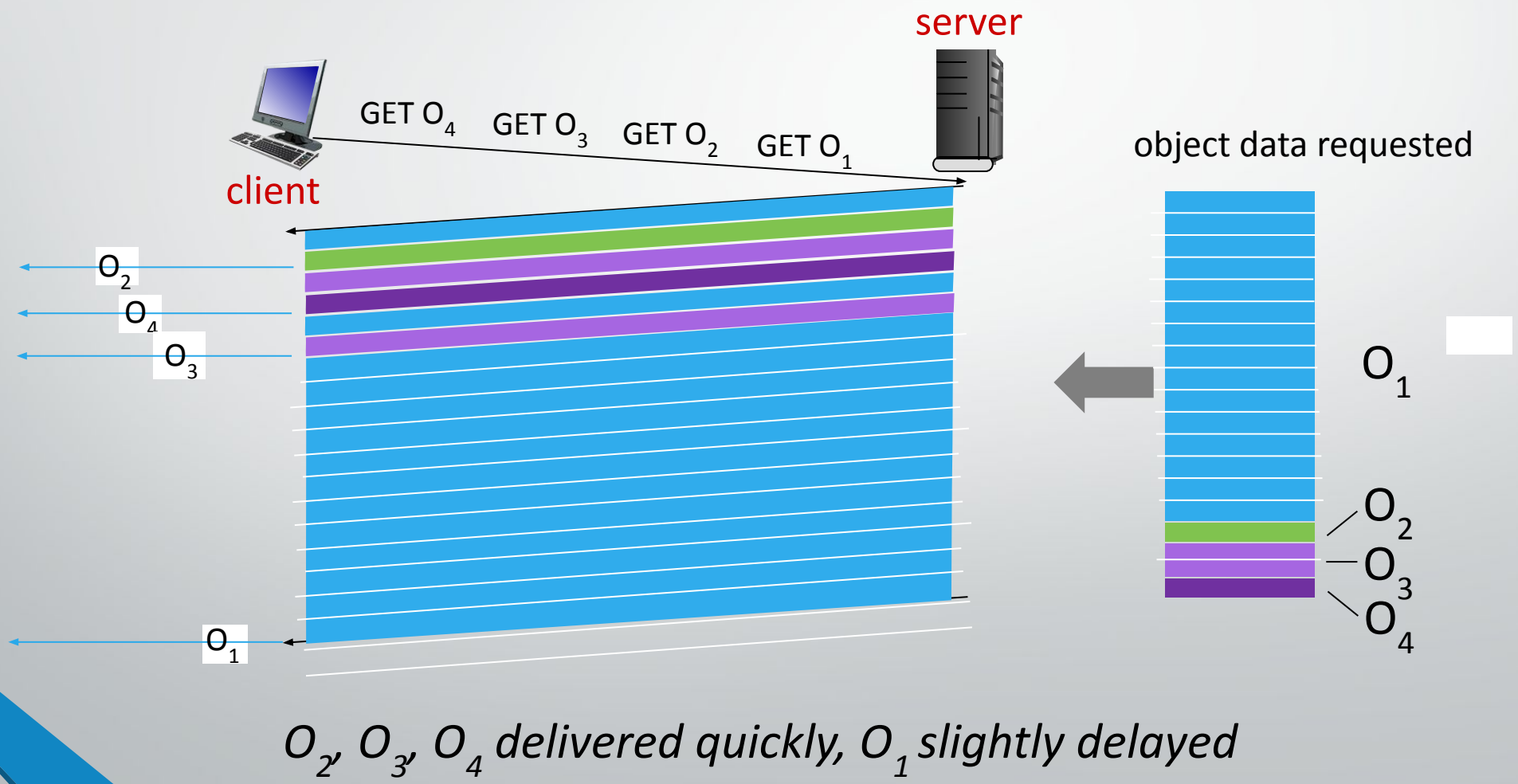
Key goal: decreased delay in multi-object HTTP requests

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

- **Multiplexing**: divide objects into frames, schedule frames to mitigate HOL blocking
- **Server Push**: Enables servers to send resources proactively; useful for assets that the server knows the client will need (e.g., stylesheets or images)
- **Stream Prioritization**: Allows clients to prioritize requests, letting the server know the order in which the client prefers to receive responses.

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

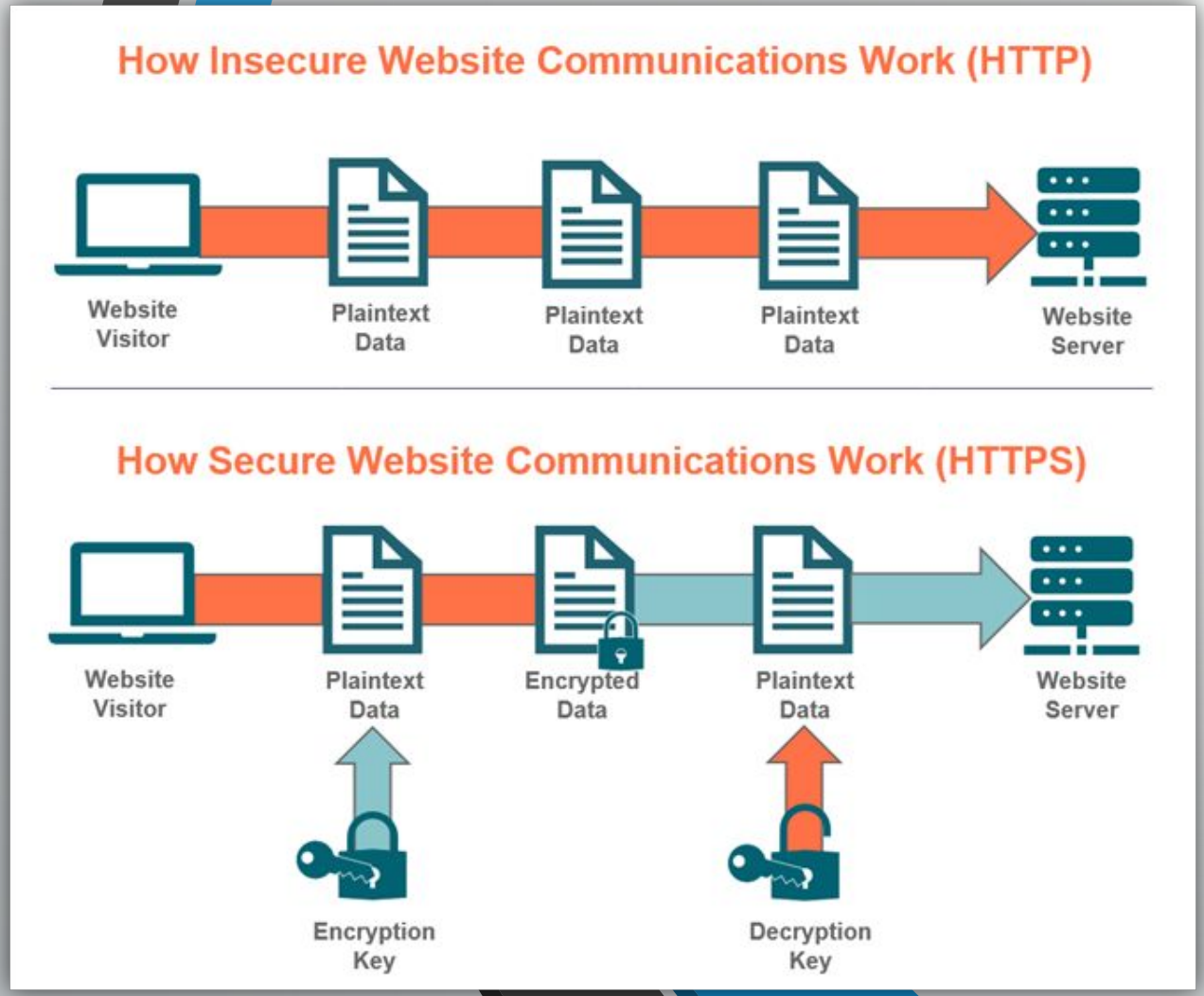
HTTP/2 over single TCP connection means:

- no security over vanilla TCP connection
- HTTP/3:
- **Built on QUIC**: Utilizes QUIC (over UDP) to manage connections, reducing latency via integrated transport and security protocols.
- **Connection Migration**: Supports changes in the client's network connection (e.g., switching from Wi-Fi to cellular) without needing to restart sessions.

HTTPS

Part 6

HTTPS



- More secure version of HTTP
- Allows transferring the data in an encrypted form.

- Uses an encryption protocol known as **Transport Layer Security**, and officially, it is referred to as a **Secure Sockets Layer (SSL).**

- HTTPS transmits the data over 443 port number.

HTTPS

- Websites to have an HTTPS protocol, need to install the **signed SSL certificate.**
- The SSL protocol encrypts the data which the client transmits to the server.
- This digital certificate is a file that contains
  - Information about the organization to help it authenticate
  - Cryptographic information that helps site users communicate with it securely through encryption.

- It is a transport layer protocol.
- **SEO Advantage** : GOOGLE gives the preferences to those websites that use HTTPS rather than the websites that use HTTP.

THE END