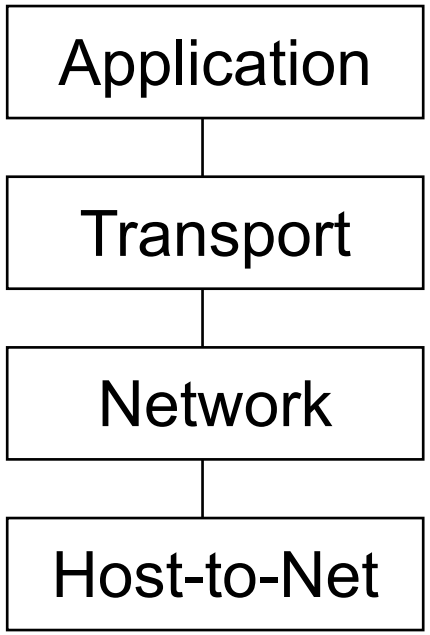
The Application Layer:
HTTP: Introduction

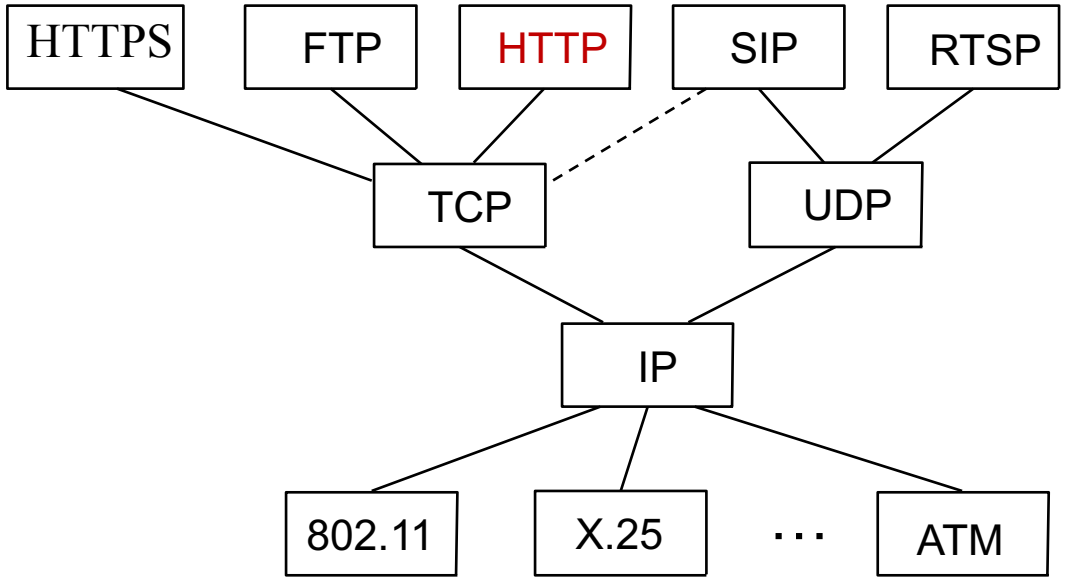
CS 352, Lecture 4.1
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

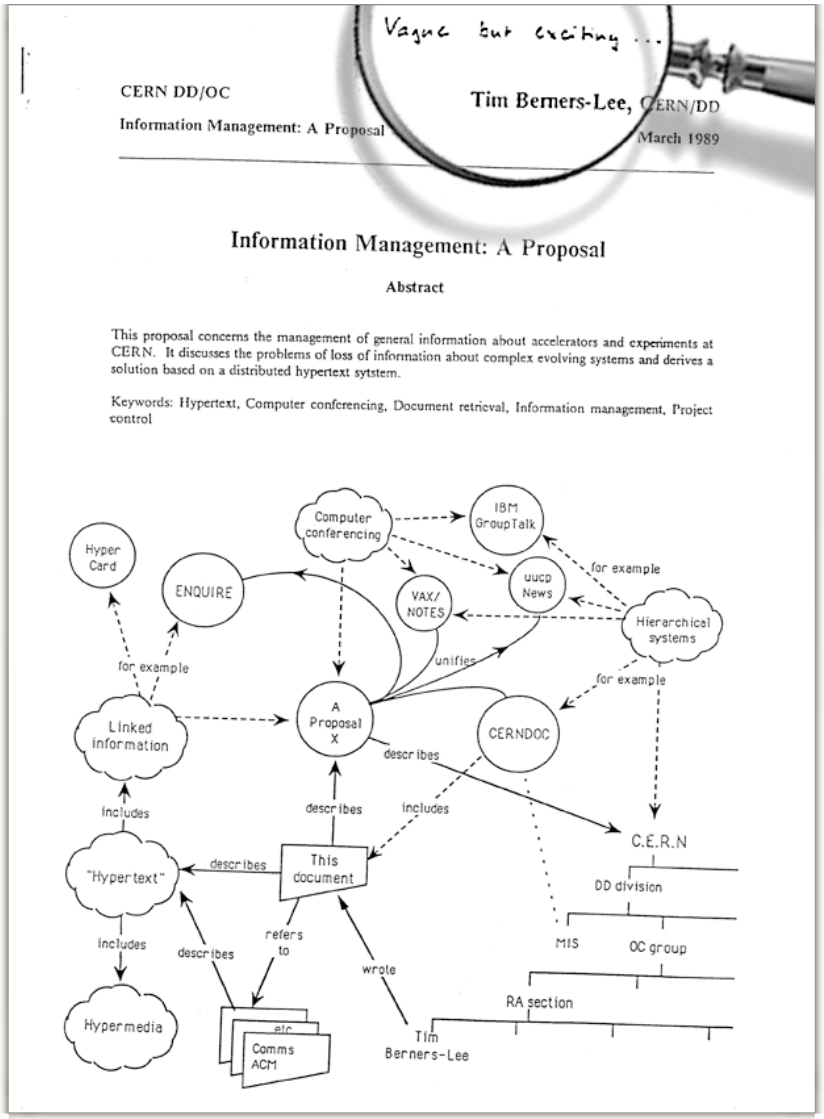
Review of concepts





- Layering and modularity; application layer
- 4-tuple (IP_s, $port_s$, IP_d, $port_d$), socket
- Client-server, peer to peer architectures

The Web: Humble origins



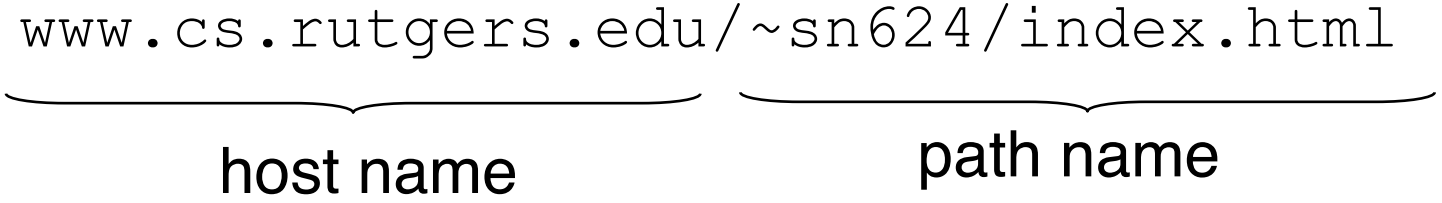
Tim Berners-Lee: a way to manage and access documents at CERN research lab

Info containing links to other info, accessible remotely, through a standardized mechanism.

His boss is said to have written on his proposal:
“vague, but exciting”

Web and HTTP: Some terms

- HTTP stands for “HyperText Transfer Protocol”
- A web page consists of many objects
- Object can be HTML file, JPEG image, video stream chunk, audio file,…
- Web page consists of base HTML-file which includes several referenced objects.
- Each object is addressable by a uniform resource locator (URL)
  - sometimes also referred to as uniform resource identifier (URI)
- Example URL:

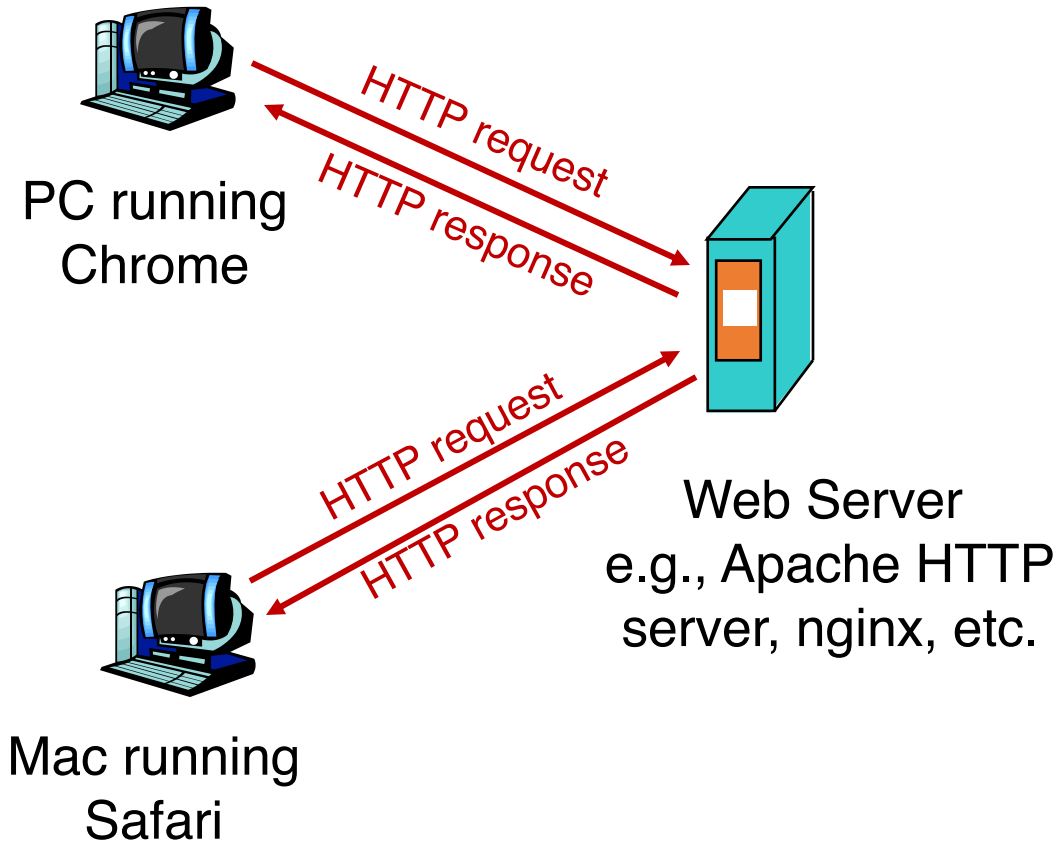


HTTP Protocol Overview

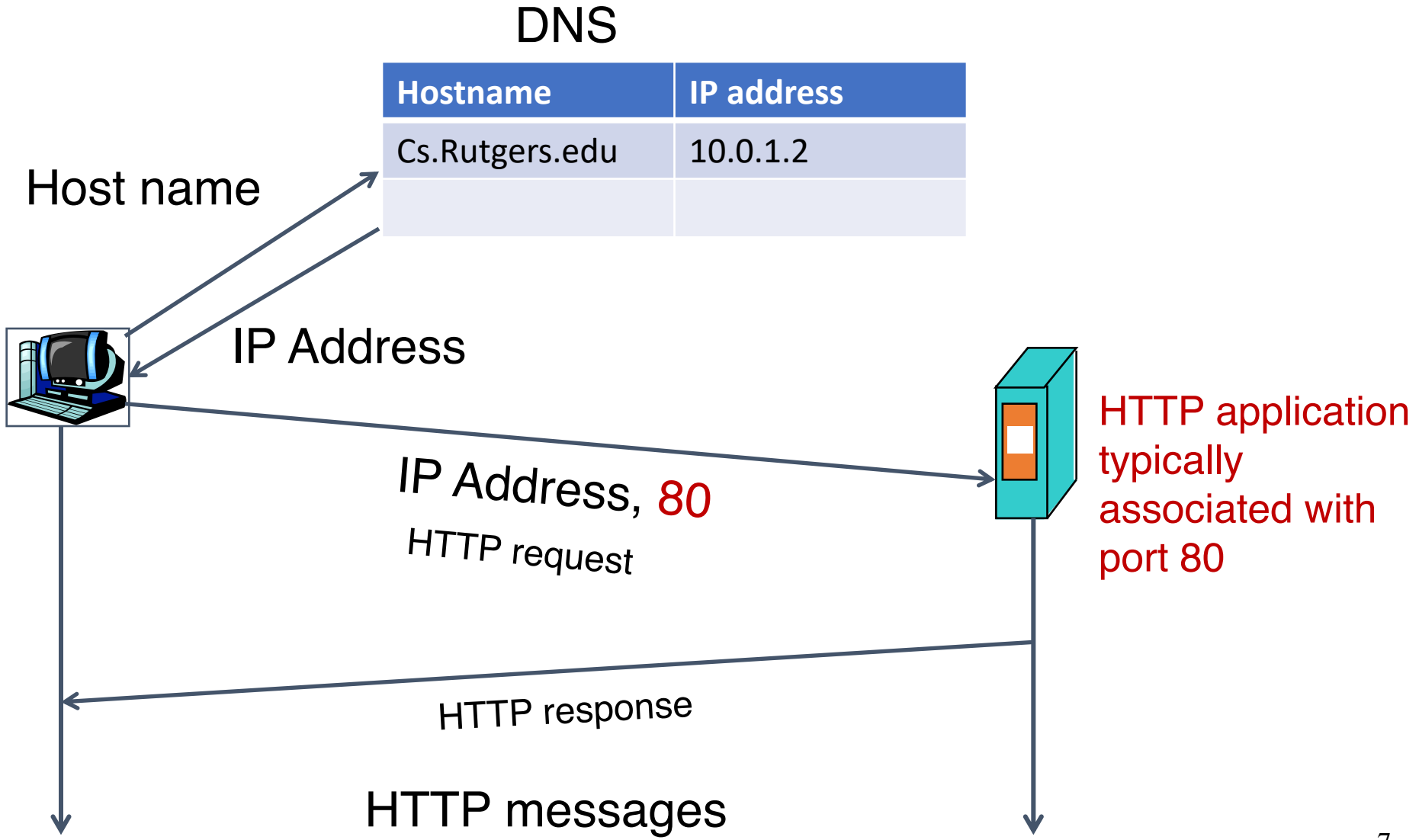
HTTP overview

HTTP: hypertext transfer protocol

- Client/server model
  - *Client:* browser that requests, receives, “displays” Web objects
  - *Server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

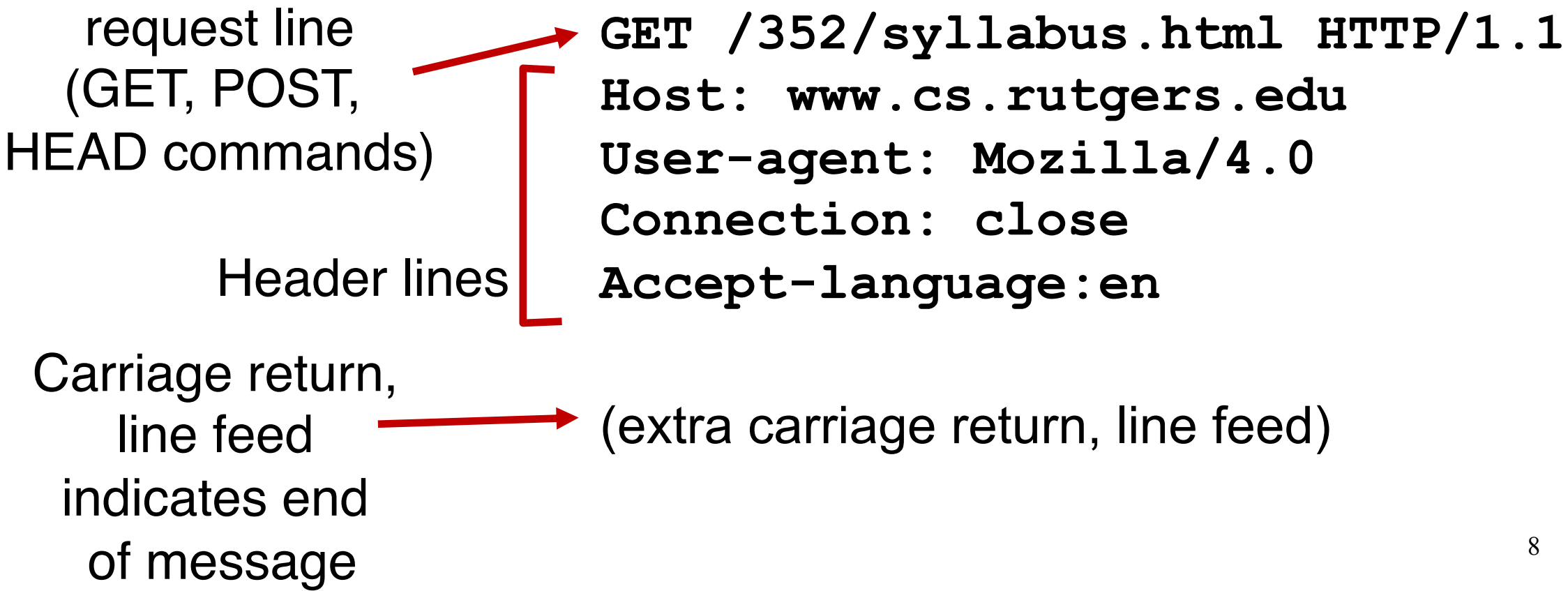


Client server connection

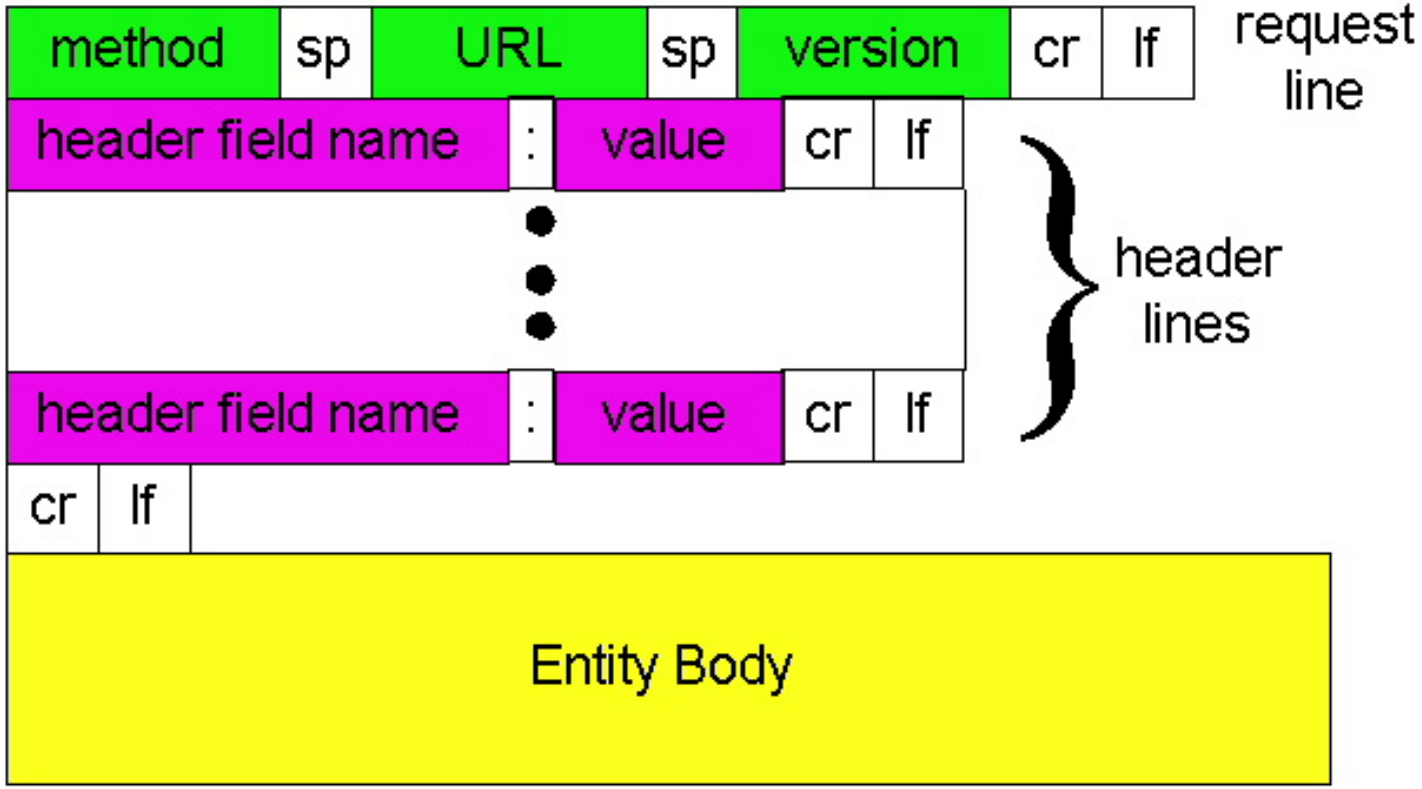


HTTP messages: request message

- ASCII (human-readable format)



HTTP Request: General format



http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14

HTTP method types

- GET
  - Get the resource specified in the requested URL. URL may refer to a data-handling process
- POST
  - Send entities (specified in the entity body) to a data-handling process at the requested URL
- HEAD
  - Asks server to leave requested object out of response, but send the rest of the response
- PUT
  - Update a resource at the requested URL with the new entity specified in the entity body
- DELETE
  - Deletes file specified in the URL
- ... and other methods.

Difference between POST and PUT

- POST: the URL of the request identifies the resource that processes the entity body
- PUT: the URL of the request identifies the resource that is contained in the entity body

https://tools.ietf.org/html/rfc2616

Difference between HEAD and GET

- GET: return the requested resource in the entity body of the response along with response headers (we'll see these shortly)
- HEAD: return all the response headers in the GET response, but without the resource in the entity body

https://tools.ietf.org/html/rfc2616

Uploading form input: GET and POST

POST method:

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

- Posted content not visible in the URL
  - Free form content (ex: images) can be posted since entity body interpreted as data bytes

GET method:

- Entity body is empty
- Input is uploaded in URL field of request line

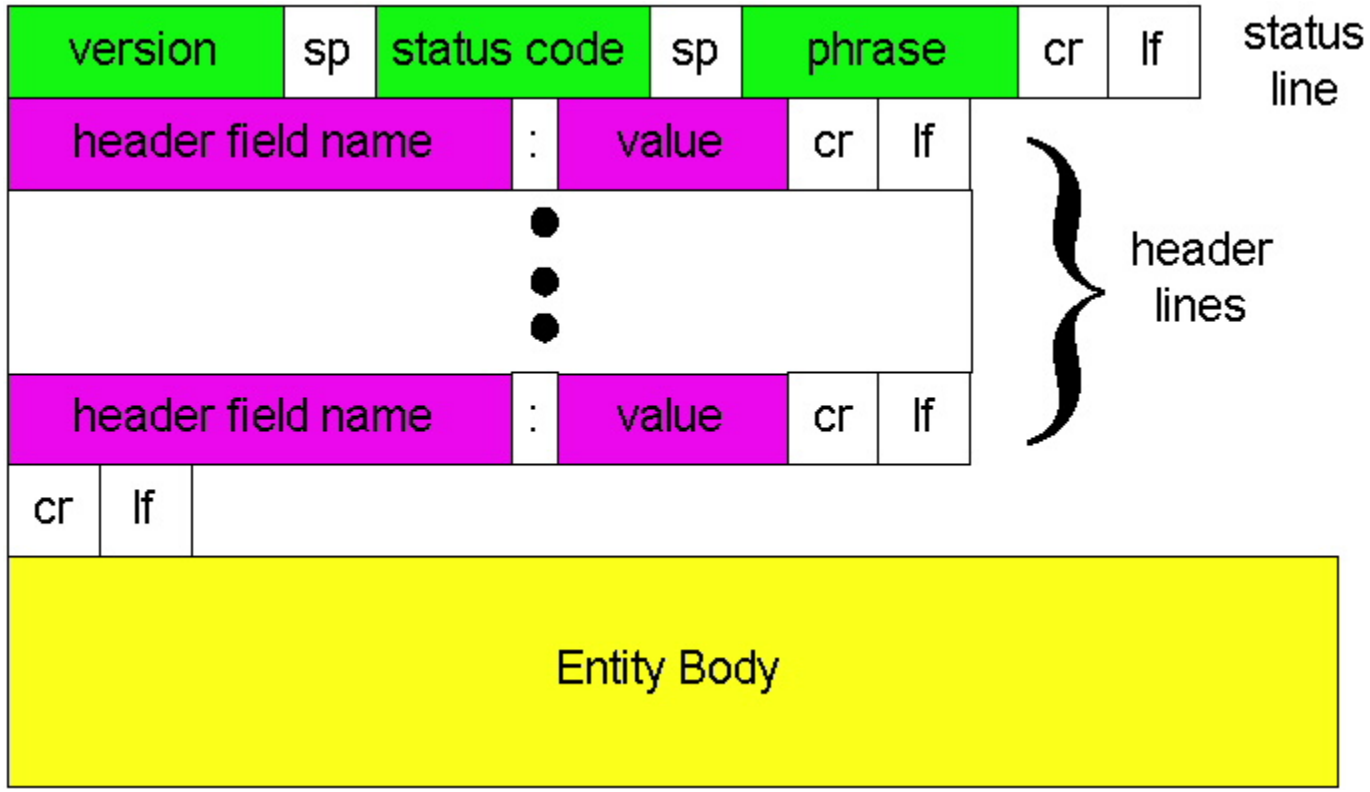
- Example:
  - http://site.com/form?first=jane&last=austen

Observing HTTP GET and POST

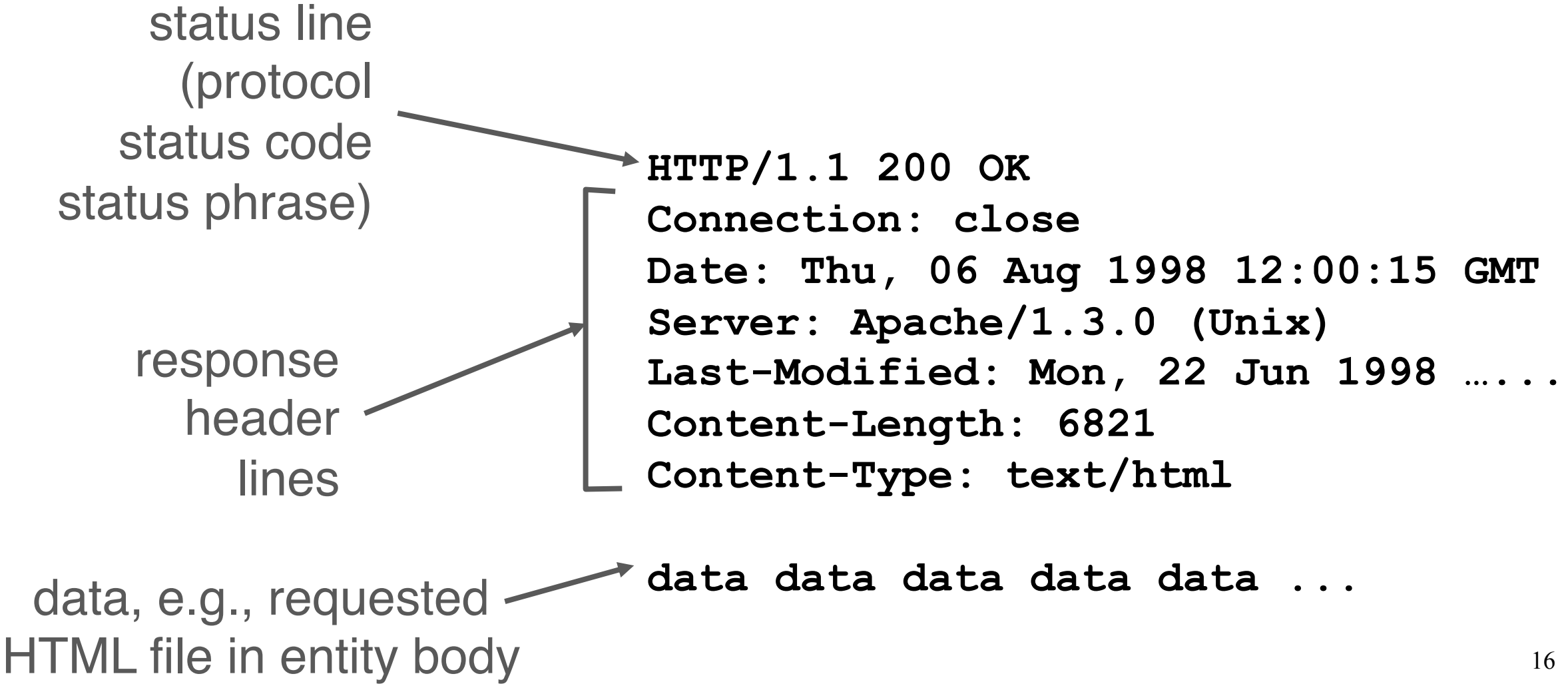
- A small demo

HTTP Response: General format

Unlike HTTP request, No method name



HTTP message: response message



HTTP response status codes

In first line in server->client response message.
A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

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

403 Forbidden

- Insufficient permissions to access the resource

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Try sending a HTTP request yourself!

1. Telnet to your favorite Web server:

```
telnet web.mit.edu 80
```

Opens TCP connection to port 80 (default HTTP server port). Anything typed in sent to port 80 at web.mit.edu

2. Type in a GET HTTP request:

```
GET / HTTP/1.1
Host: web.mit.edu
```

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

HTTP: Persistence, Cookies, and Caching

CS 352, Lecture 4.2

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Additional details about HTTP

- Persistent vs. Nonpersistent HTTP connections
- Cookies (User-server state)
- Web caches

Non/Persistent HTTP

HTTP connections

Non-persistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

TCP is a kind of reliable communication service provided by the transport layer. It requires the connection to be “set up” before data communication.

Non-persistent HTTP

Suppose user visits a page with text and 10 images.

1a. HTTP client initiates TCP connection to HTTP server

1b. HTTP server at host "accepts" connection, notifying client

2. HTTP client sends HTTP request message

3. HTTP server receives request message, replies with response message containing requested object

time

Non-persistent HTTP (contd.)

4. HTTP server closes TCP connection.

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

time

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

Non-Persistent HTTP's Response time

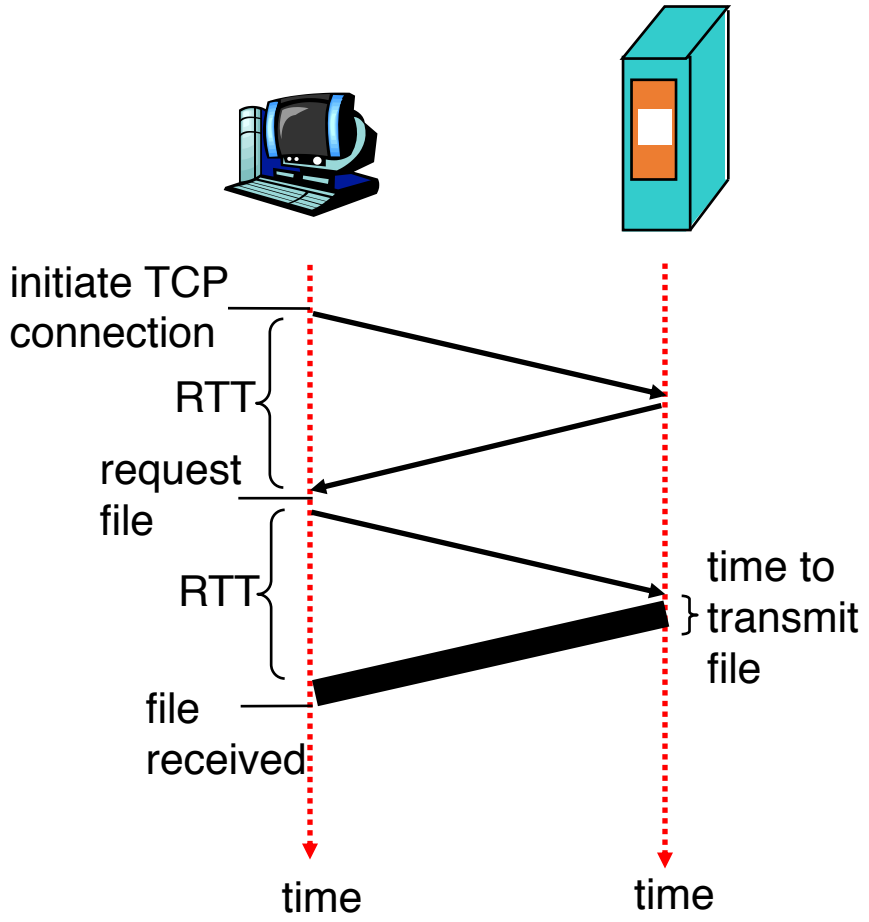
Round Trip Time (RTT): time to send a packet from client to server and back.

- Sum of propagation, transmission, and queueing delays along both directions.

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

total = 2RTT + file transmission time



Persistent HTTP: jumping to steps 4/5

4. HTTP server sends a response.
Server keeps the TCP connection alive.

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

time

The 10 objects can be requested over the same TCP connection.

i.e., save an RTT trying to open a new TCP connection per object.

Persistent vs. Non-persistent

Non-persistent HTTP requires at least 2 RTTs per object.

- For each object: Open TCP connection; send HTTP request & receive response

Persistent HTTP: since server leaves connection open after sending response, subsequent HTTP messages between same client/server sent over the open connection

Save one RTT per object relative to non-persistent HTTP

In both cases, browsers have a choice of opening multiple parallel connections.

Remembering HTTP users

HTTP: User data on servers?

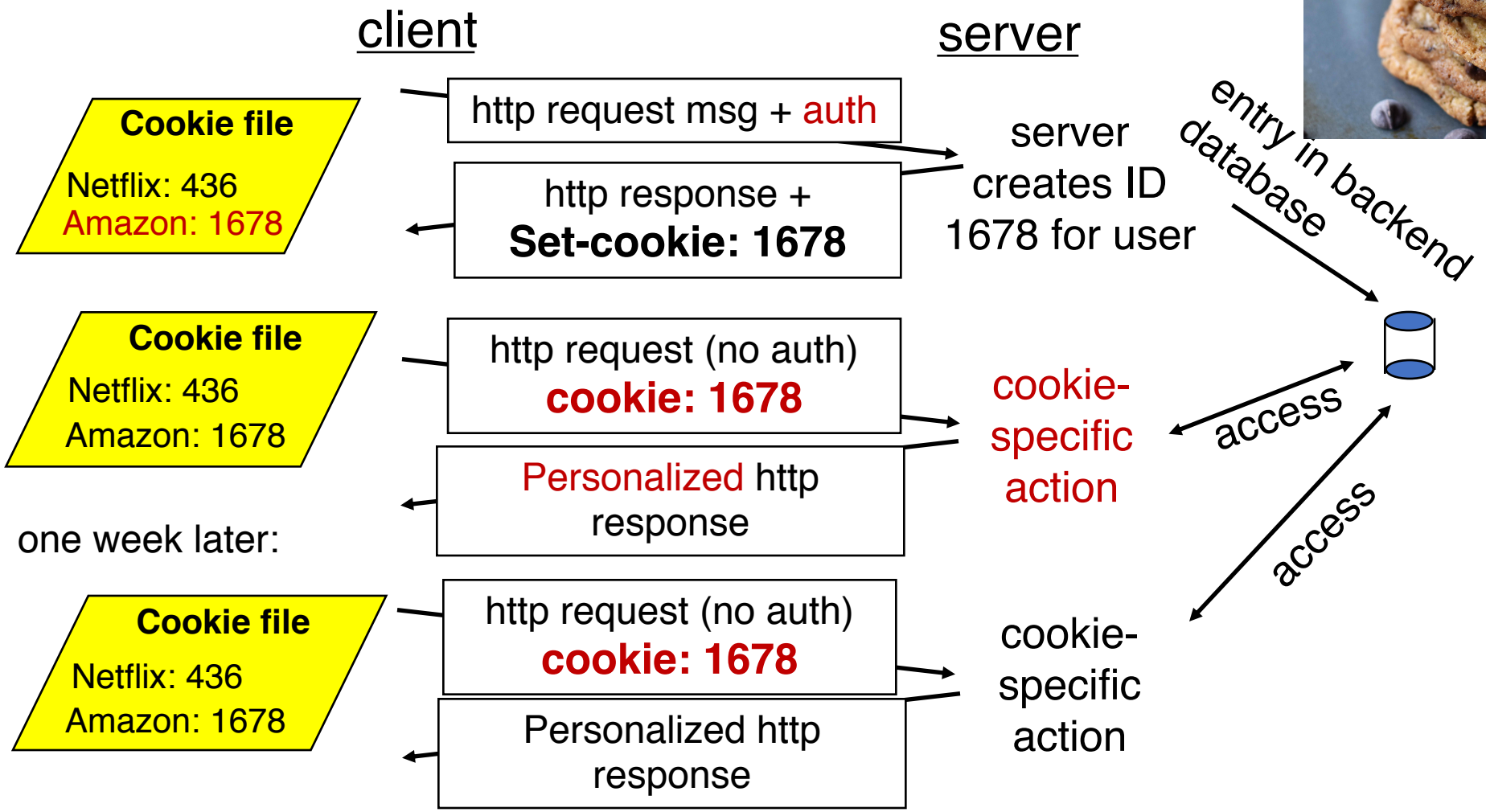
So far, HTTP is "stateless"

- The server maintains no memory about past client requests

But state, i.e., memory, about the user at the server is very useful!

- authorization
- shopping carts
- recommendations
- In general, user session state

Cookies: Keeping user memory



How cookies work

Four components:

1. cookie header line of HTTP response message
2. cookie header line in HTTP request message
3. cookie file kept on user endpoint, managed by user's browser
4. back-end database maps cookie to user data at Web endpoint

Client and server collaboratively track and remember the user's state.

PSA: Cookies and Privacy

- The Internet would be unusable without cookies.
- However, cookies permit sites to learn a lot about you, from your behaviors.
- E.g., which products, topics, images, etc. are you most interested in?
  - What demographic do you belong to? Where?
  - What kinds of ads will you likely click on?
  - Tracking networks correlate this info across sites
- You might reasonably be concerned about your privacy when online.
- Disable and delete unnecessary cookies by default
- Use privacy-conscious browsers, websites, tools: DuckDuckGo, Brave, AdBlock Plus.

Caching in HTTP

Web caches

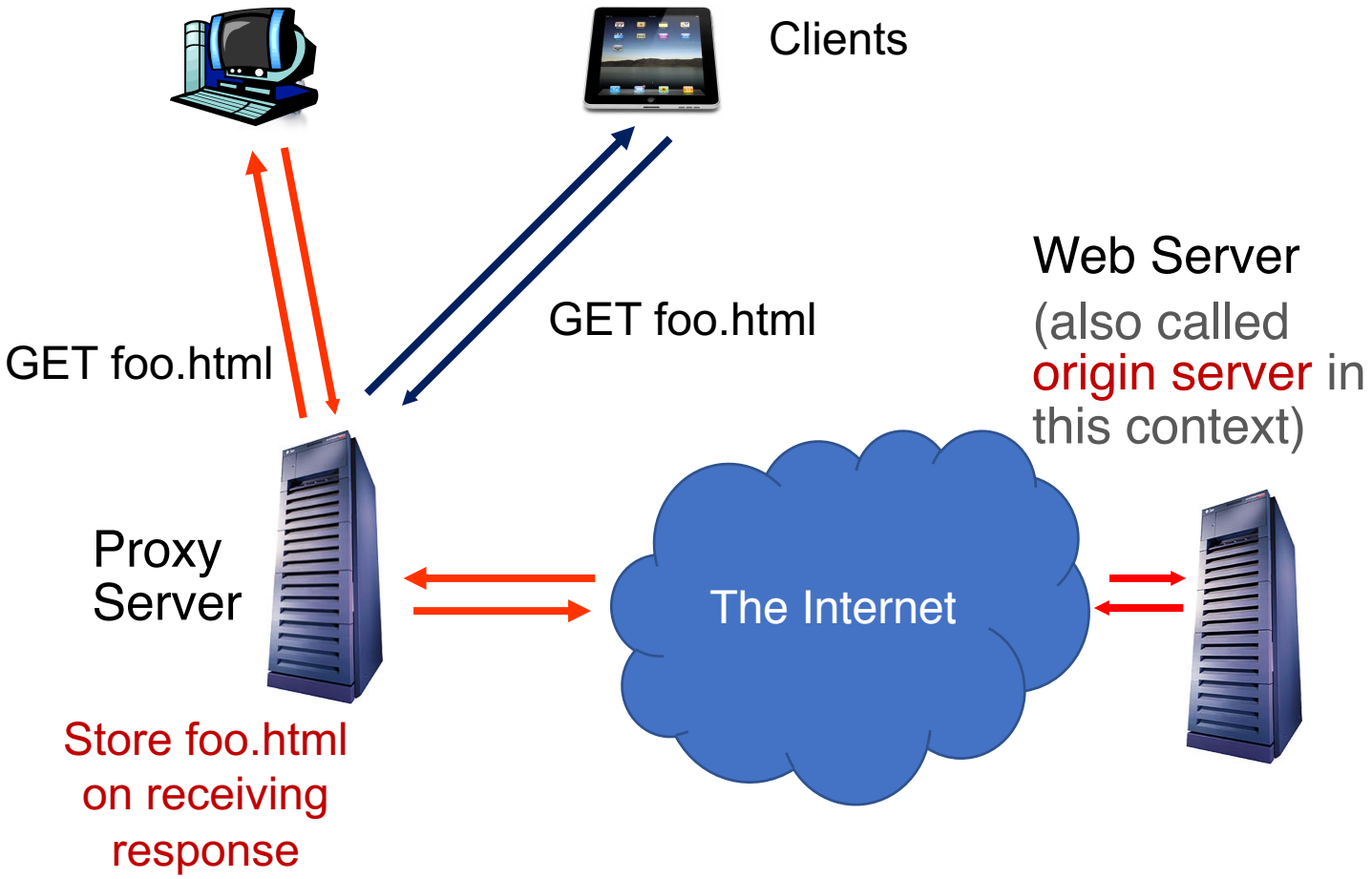
Web caches: Machines that remember web responses for a network

Why cache web responses?

- Reduce response time for client requests
- Reduce traffic on an institution's access link

Caches can be implemented in the form of a proxy server

Web caching using a proxy server

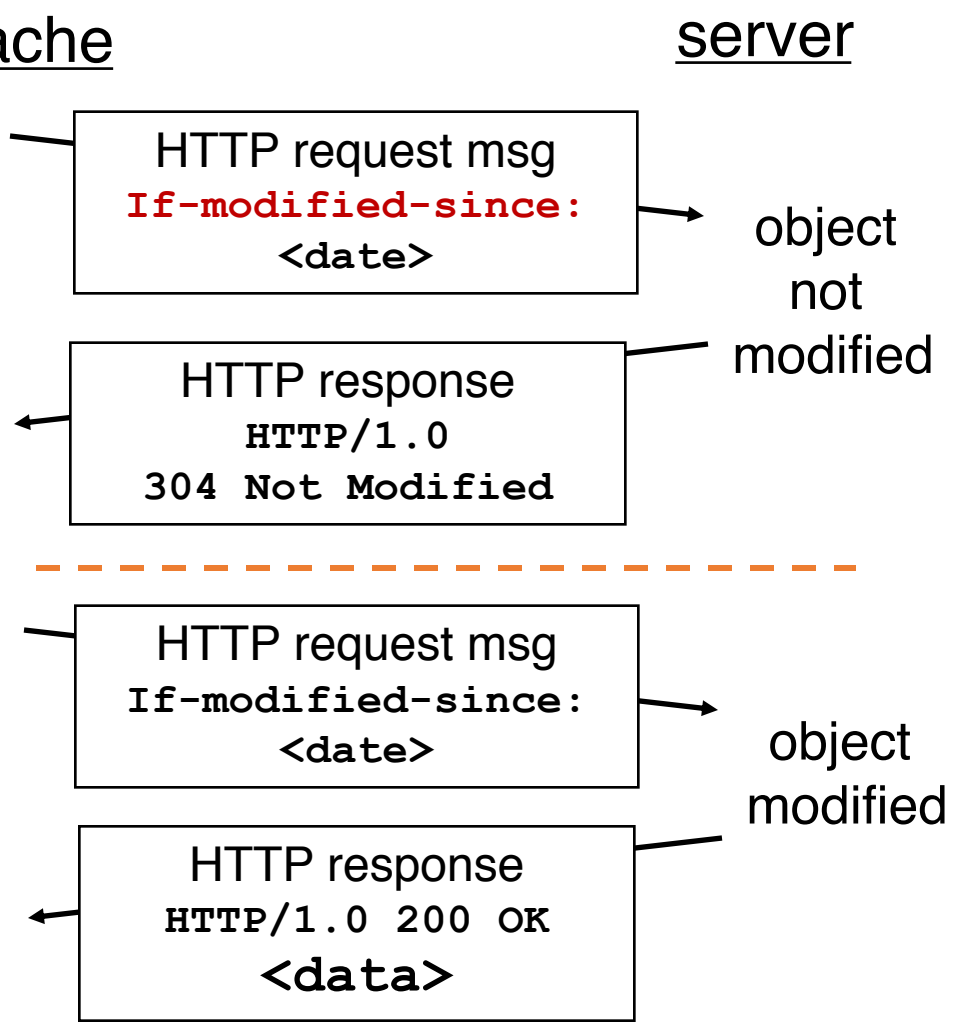


- You can configure a HTTP proxy on your laptop's network settings.
- If you do, your browser sends all HTTP requests to the proxy (cache).
- Hit: cache returns object
- Miss:
  - cache requests object from origin server
  - caches it locally
  - and returns it to client

Web Caches: how does it look on HTTP?

- Conditional GET guarantees cache content is up-to-date while still saves traffic and response time whenever possible

- Date in the cache's request is the last time the server provided in its response header "last modified"



Content Distribution Networks (CDN)

A global network of web caches

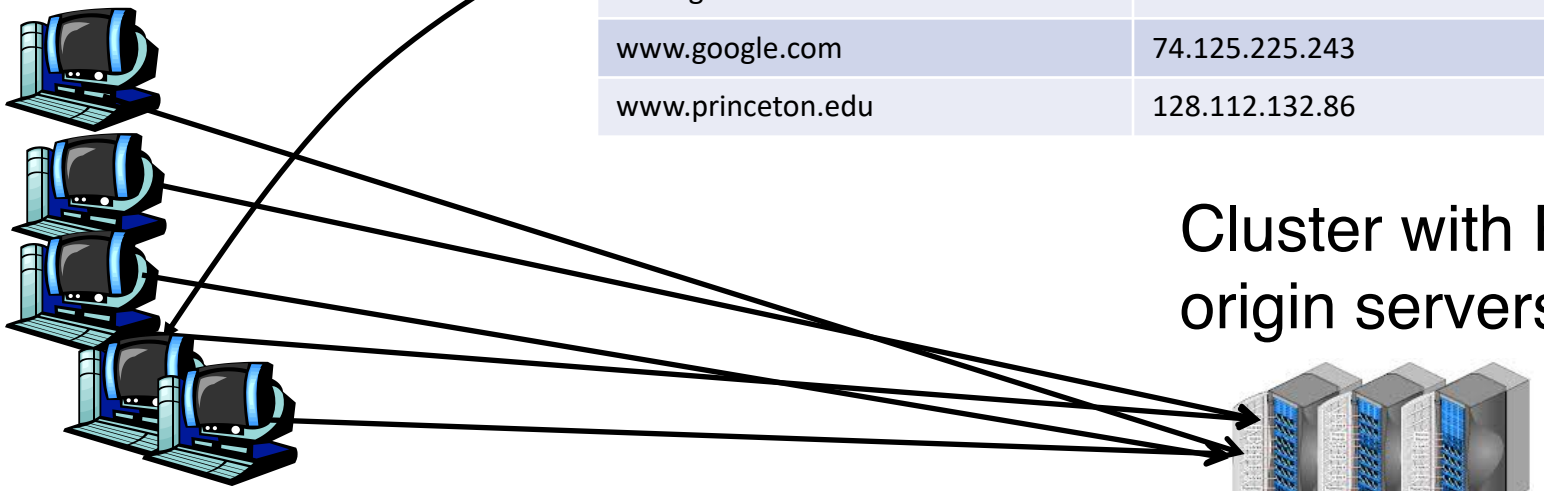
- Provisioned by ISPs and network operators
- Or content providers, like Netflix, Google, etc.

Uses

- Reduce bandwidth requirements on content provider
- Reduce $$ to maintain origin servers
- Reduce traffic on a network's Internet connection, e.g., Rutgers
- Improve response time to user for a service

Without CDN

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



Cluster with Rutgers CS origin servers

128.6.4.2

- Huge bandwidth requirements for Rutgers
- Large propagation delays to reach users
- So, distribute content to geographically distributed cache servers.
- Often, use DNS to redirect request to users to copies of content

CDN terms

- Origin server
  - Server that holds the authoritative copy of the content
- CDN server
  - A replica server owned by the CDN provider
- CDN name server
  - A DNS like name server used for redirection
- Client

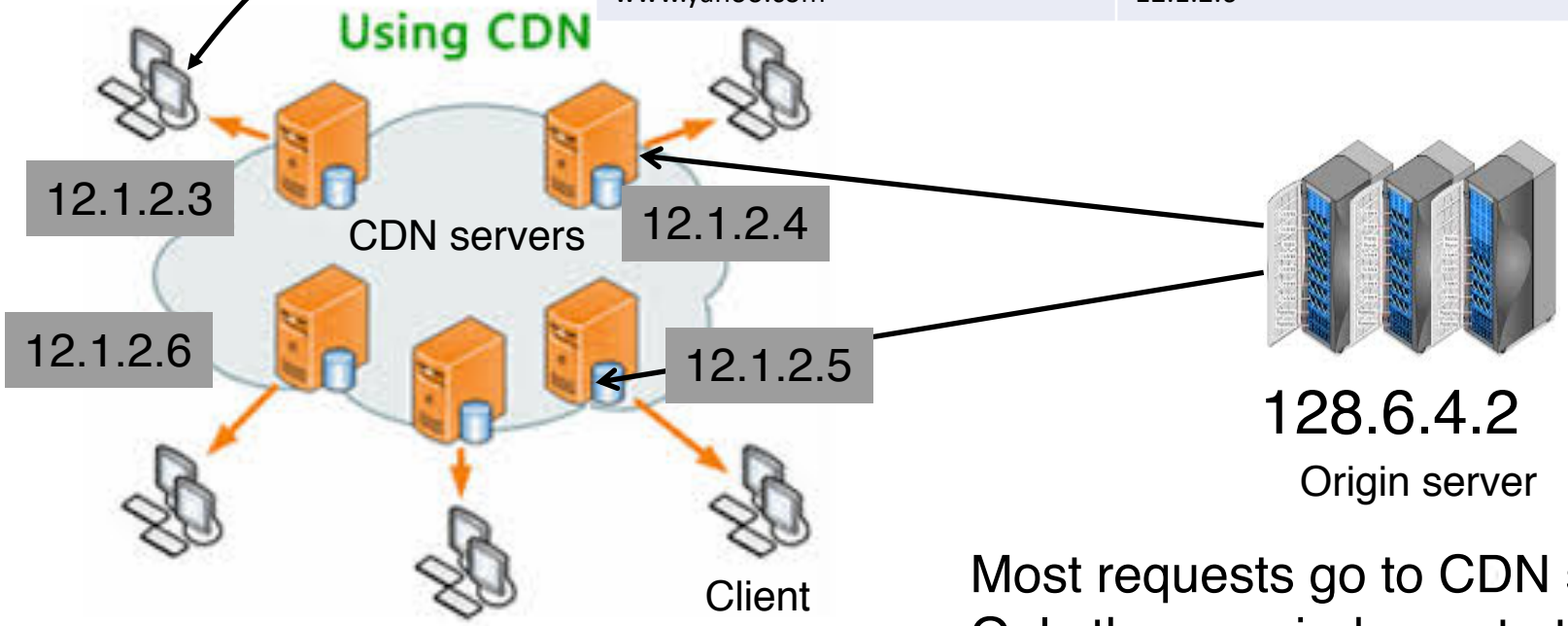
With CDN

Implement DNS delegation to the CDN name server. "layer of indirection"

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	124.8.9.8 (NS of CDN)
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
www.yahoo.com	12.1.2.3
www.yahoo.com	12.1.2.4
www.yahoo.com	12.1.2.5
www.yahoo.com	12.1.2.6



Custom logic to map ONE domain name to one of many IP addresses!

Most requests go to CDN servers (caches). Only the remainder go to the origin server.

Themes from HTTP

- Request/response nature of the protocol
  - Special HTTP headers to customize actions of the protocol
- ASCII-based message structures
- Improve performance using caching
- Scale using a layer of indirection
- Simple, highly-customizable protocol, permitting efficient implementations and flexible functionality.
  - The basis of why we enjoy the web today.