The Application Layer: HTTP

CS 352, Lecture 4

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

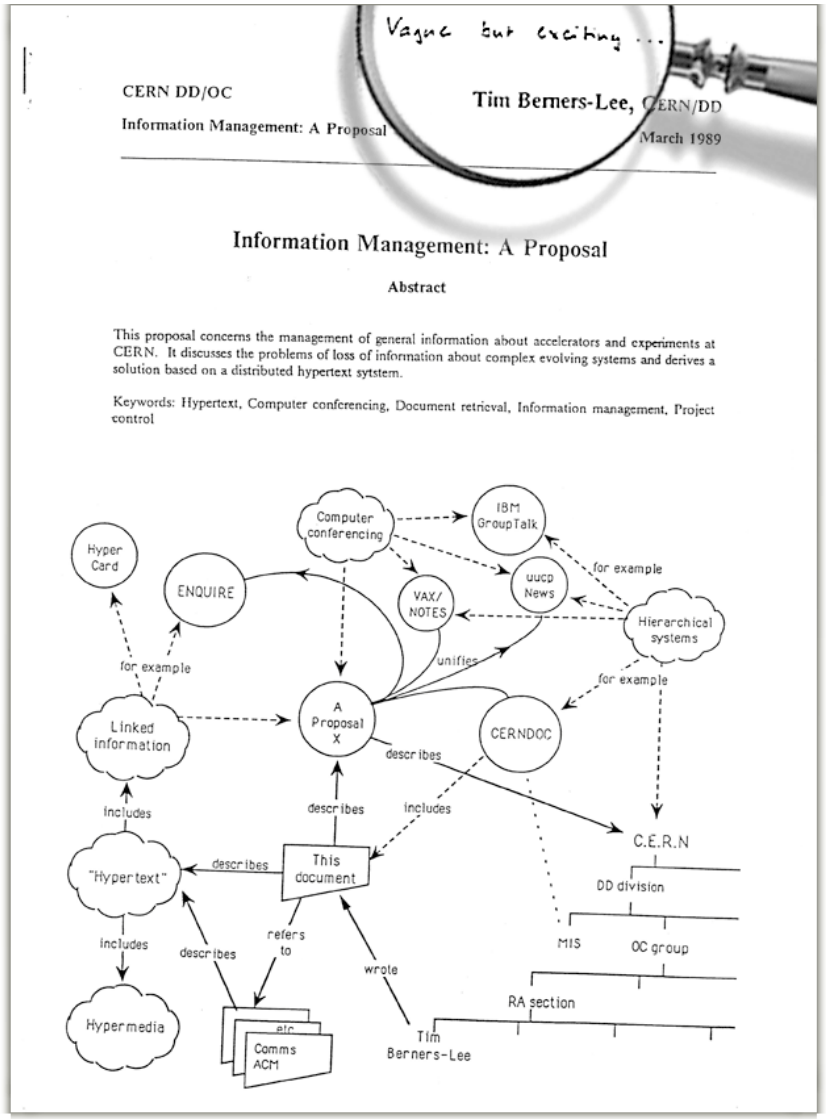
A recap: Domain Name Service (DNS)

- Hostname to IP address translation
- Hierarchical structure to scale lookups
- Recursive and Iterative queries
- Caching for performance optimization
- Multiple layers of indirection to delegate the lookup work

Some themes from DNS

- Request/response nature of protocols
- ASCII-based message structures
  - DNS, HTTP, SMTP, FTP - simple (ASCII) protocols
- Higher performance using caching
- Scale using indirection

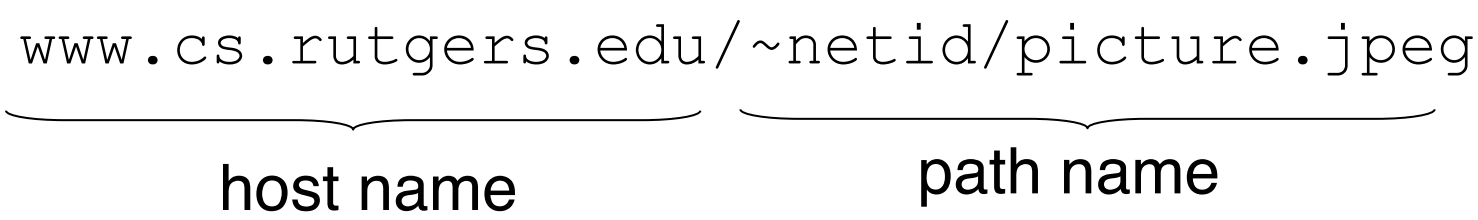
The Web



"Vague, but exciting"

Web and HTTP: Some terms

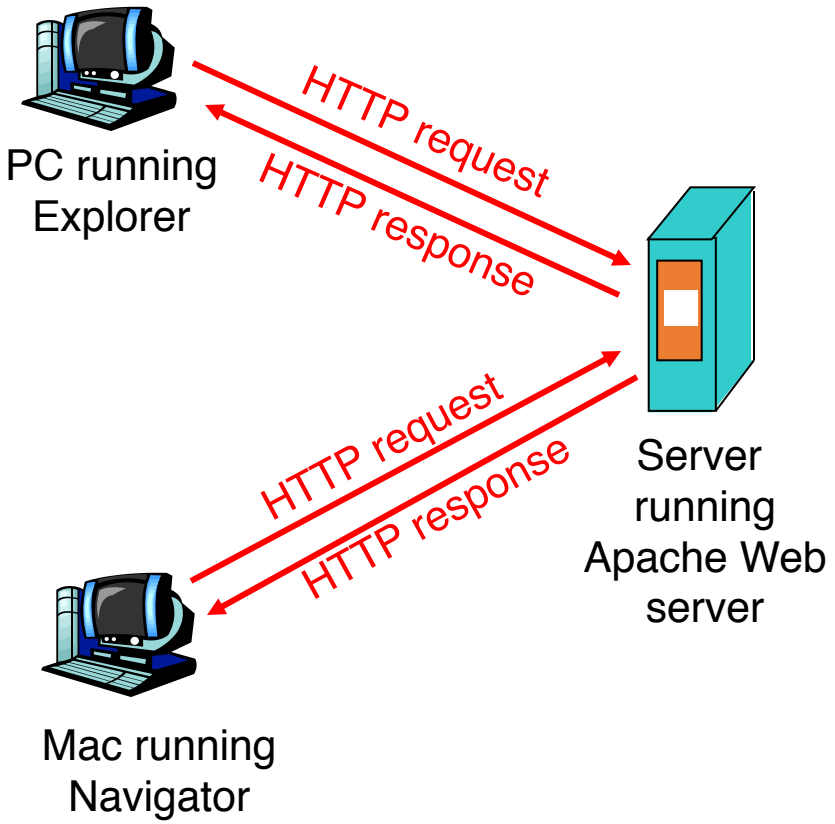
- Web page consists of objects
- Object can be HTML file, JPEG image, video stream chunk, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:



HTTP overview

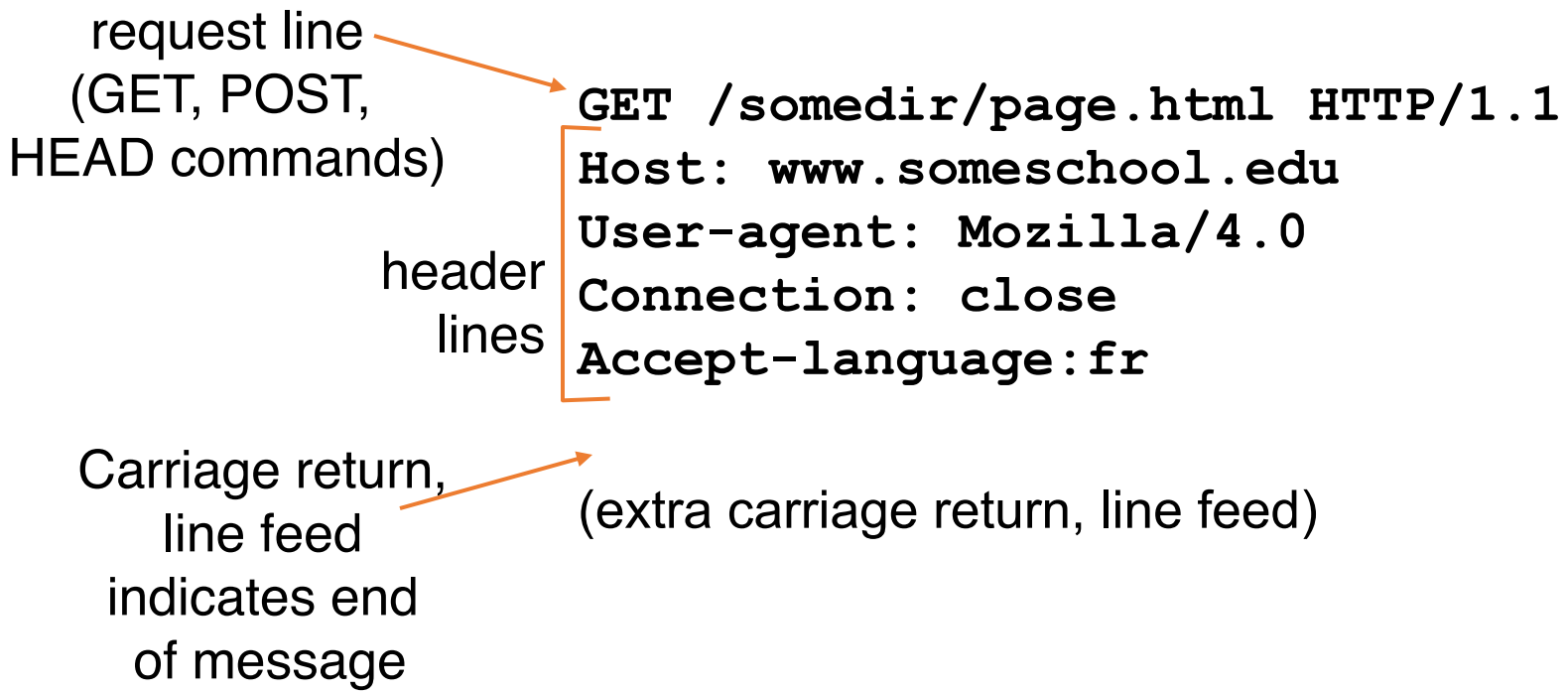
HTTP: hypertext transfer protocol

- 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

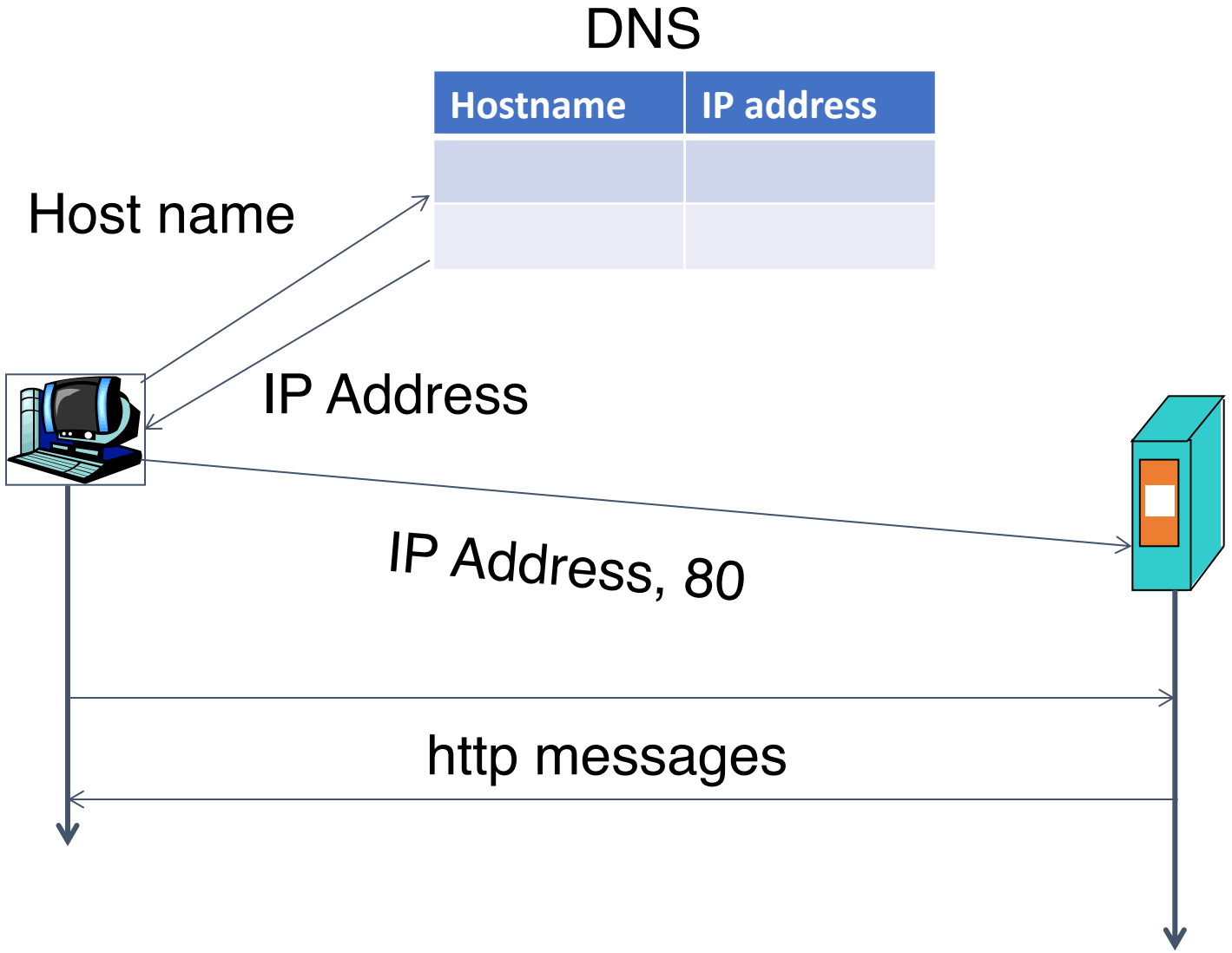


HTTP messages: request message

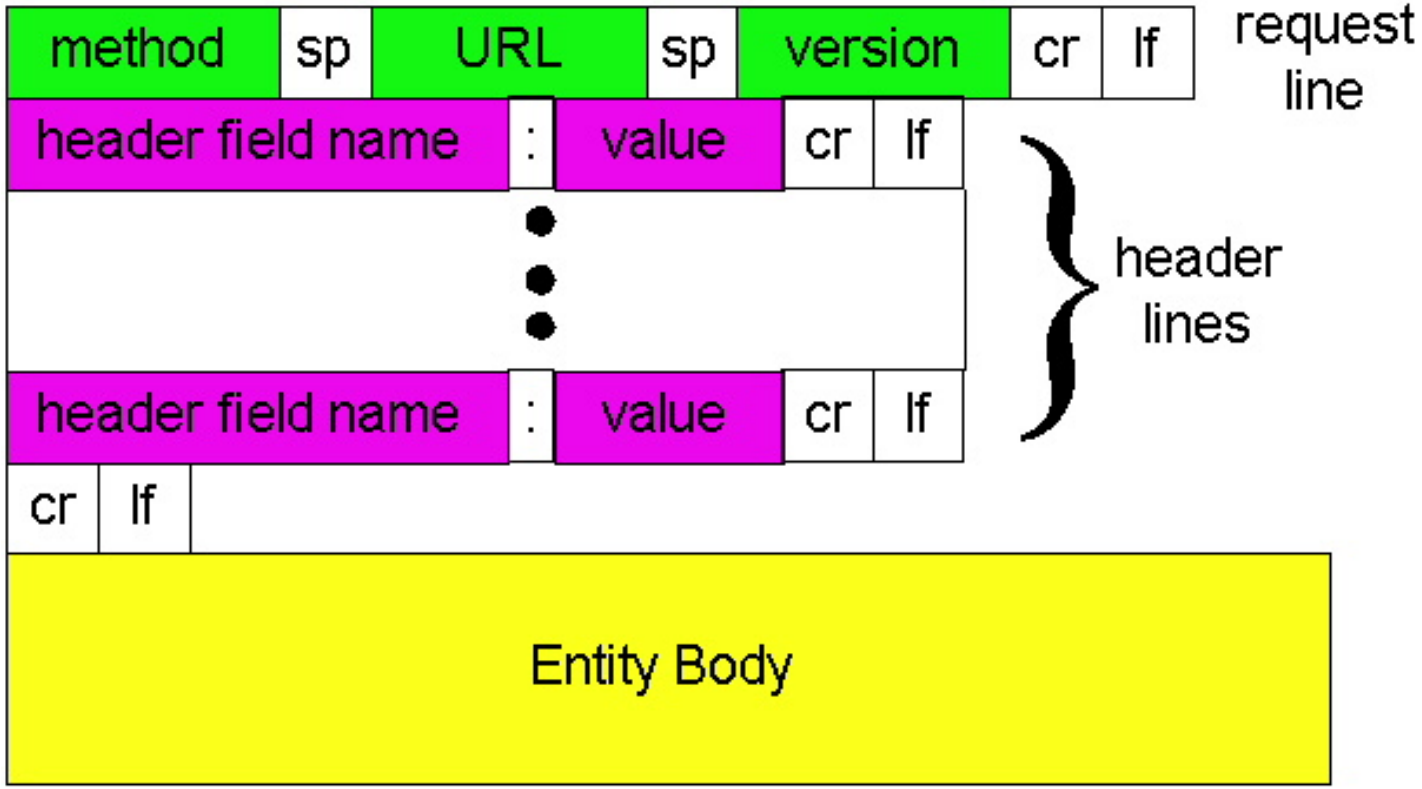
- HTTP request message:
  - ASCII (human-readable format)



Client server connection



HTTP request message: general format



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

Method types

- GET
  - Get the file specified in the path URL field in entity body
- POST
  - accept the entity enclosed in the entity body as a new subordinate of the resource identified by the URL field
- HEAD
  - asks server to leave requested object out of response
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

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=doe

Example: Client POST request

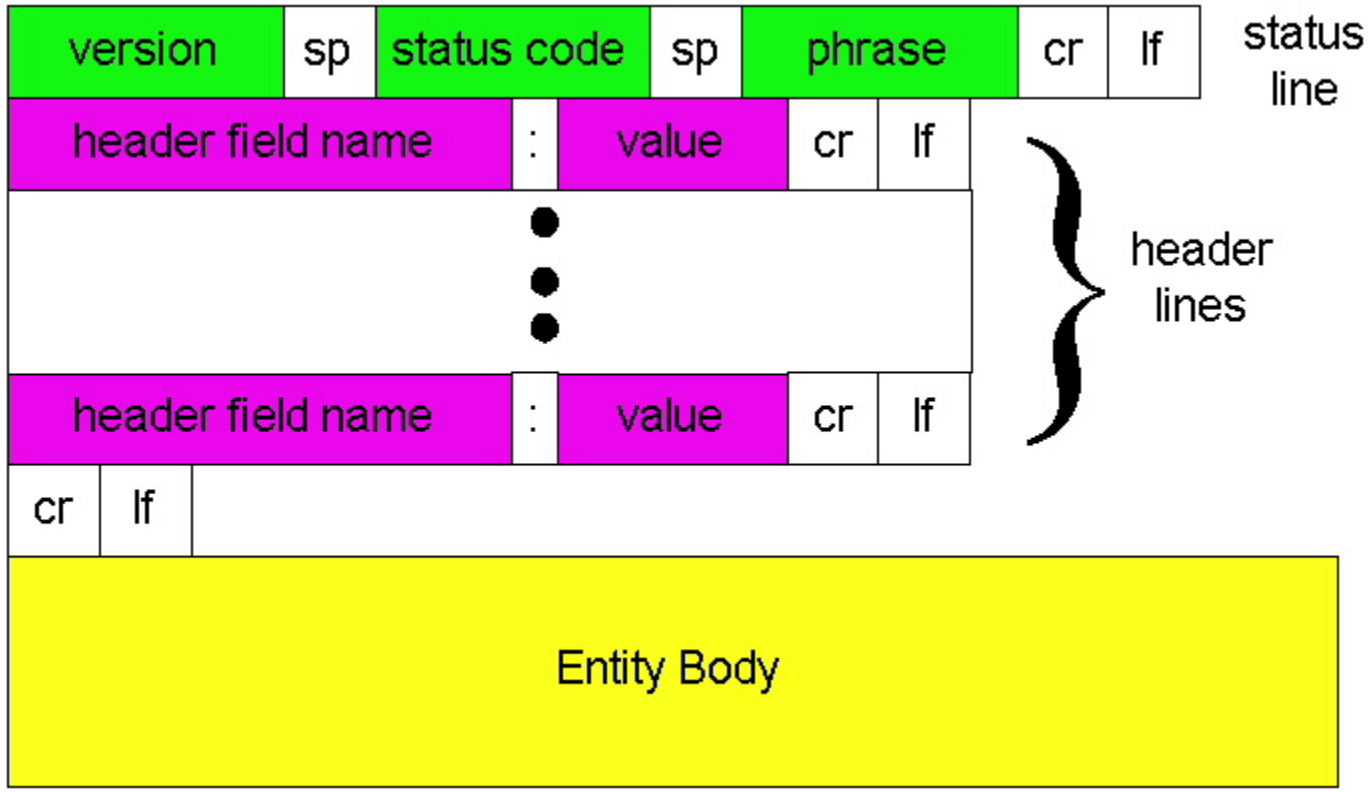
```
POST /cgi-bin/rats.cgi HTTP/1.0
Referer: http://nes:8192/cgi-bin/rats.cgi
Connection: Keep-Alive
User-Agent: Mozilla/4.73 [en] (X11; U; Linux 2.2.12-20 i686)
Host: nes:8192
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 93

Account=cs111fall&First=Alice&Last=White&SSN=123456789&Bday=01011980&State=CreateAccount
```

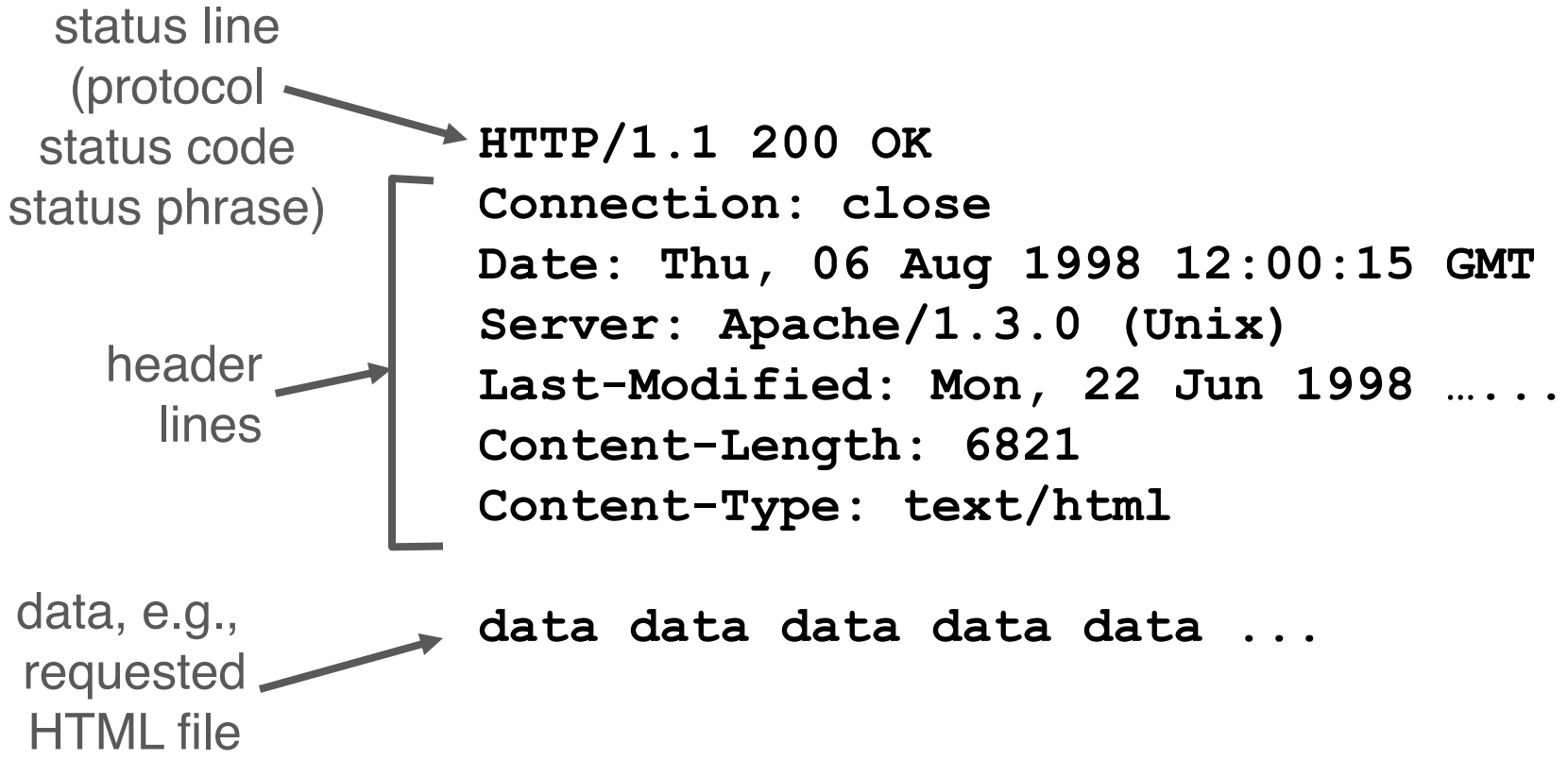
http://**www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14**

HTTP response message: 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:)

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Try out HTTP (client side) for 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 www.eden.rutgers.edu

2. Type in a GET HTTP request:

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

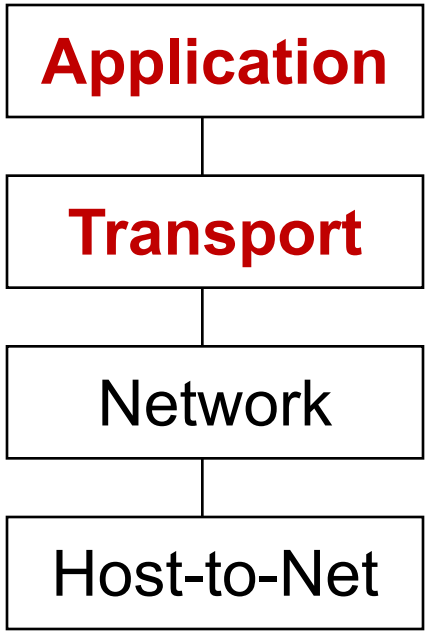
By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

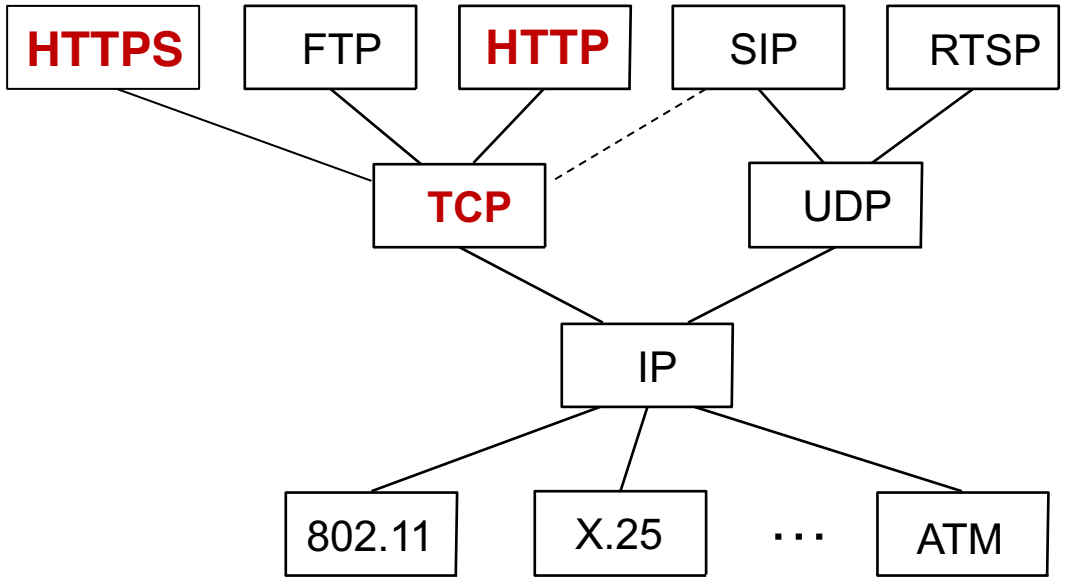
3. Look at response message sent by HTTP server!

Additional details about HTTP

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

Recall the Internet protocol stack…





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.

Web Server

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.)

Web Server

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

HTTP Response time

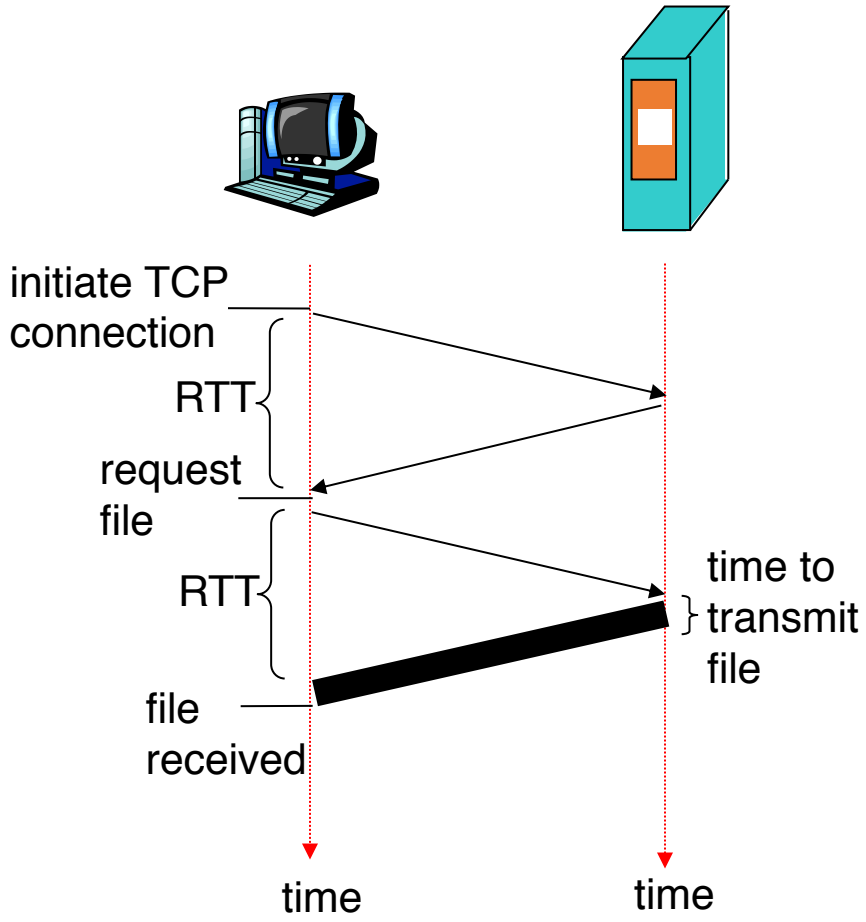
Definition of RTT: time to send a small packet to travel from client to server and back.

- Sum of propagation and queueing delays.

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 + transmit time



Persistent vs. Non-persistent

Non-persistent HTTP issues:

- requires 2 RTTs per object
- Browsers can 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

HTTP: User data on servers?

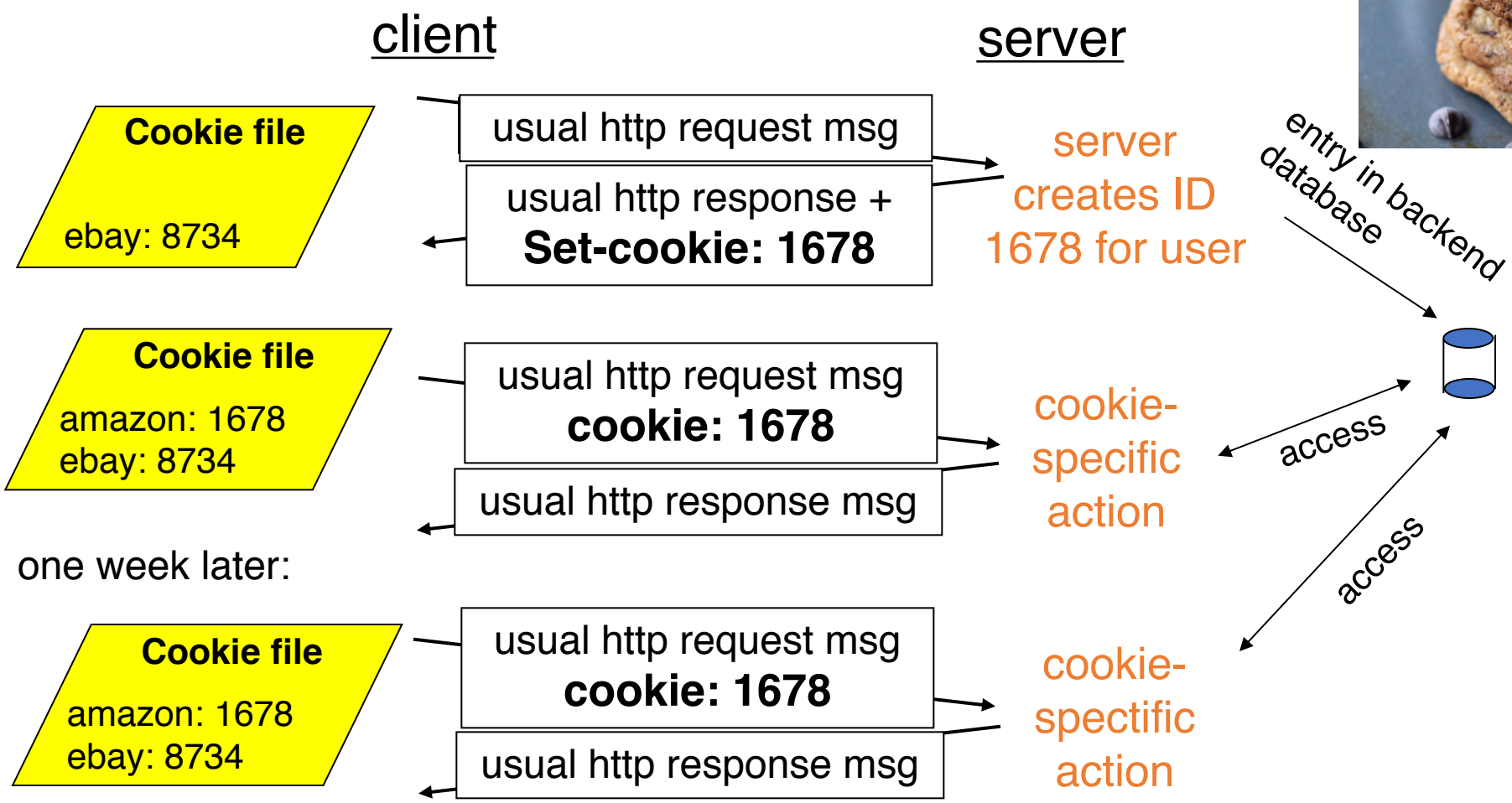
So far, HTTP is "stateless"

- The server maintains no information about past client requests

What can state about the user @ the server bring?

- authorization
- shopping carts
- recommendations
- user session state

Cookies: Keeping user memory



Summary of cookies

Four components:

1. cookie header line of HTTP *response* message

2. cookie header line in HTTP *request* message

3. cookie file kept on user's host, managed by user's browser

4. back-end database at Web site

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

Cookies and Privacy

Aside

Cookies and privacy

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

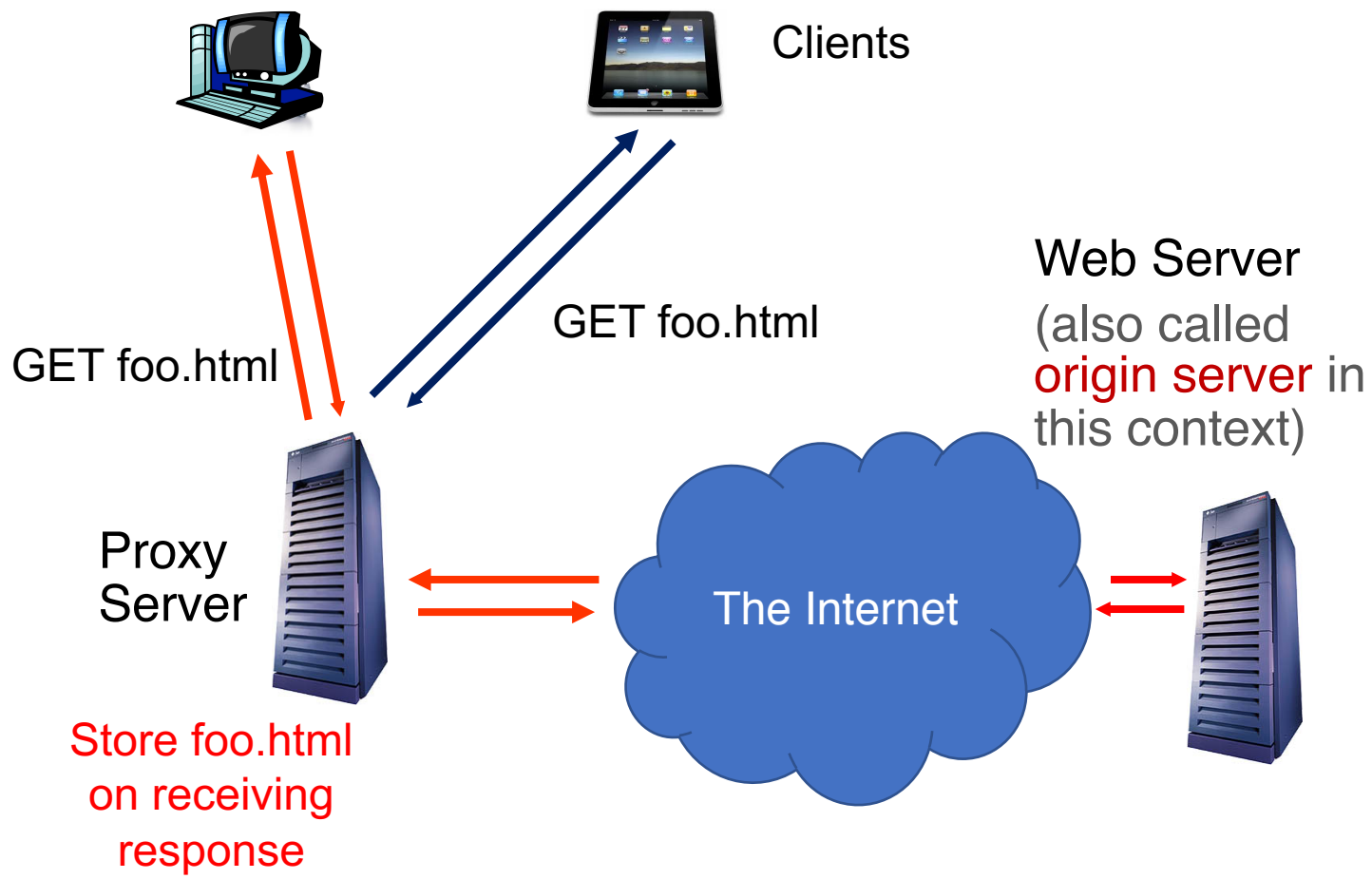
Web caches (proxy server)

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

Web caches (proxy server)

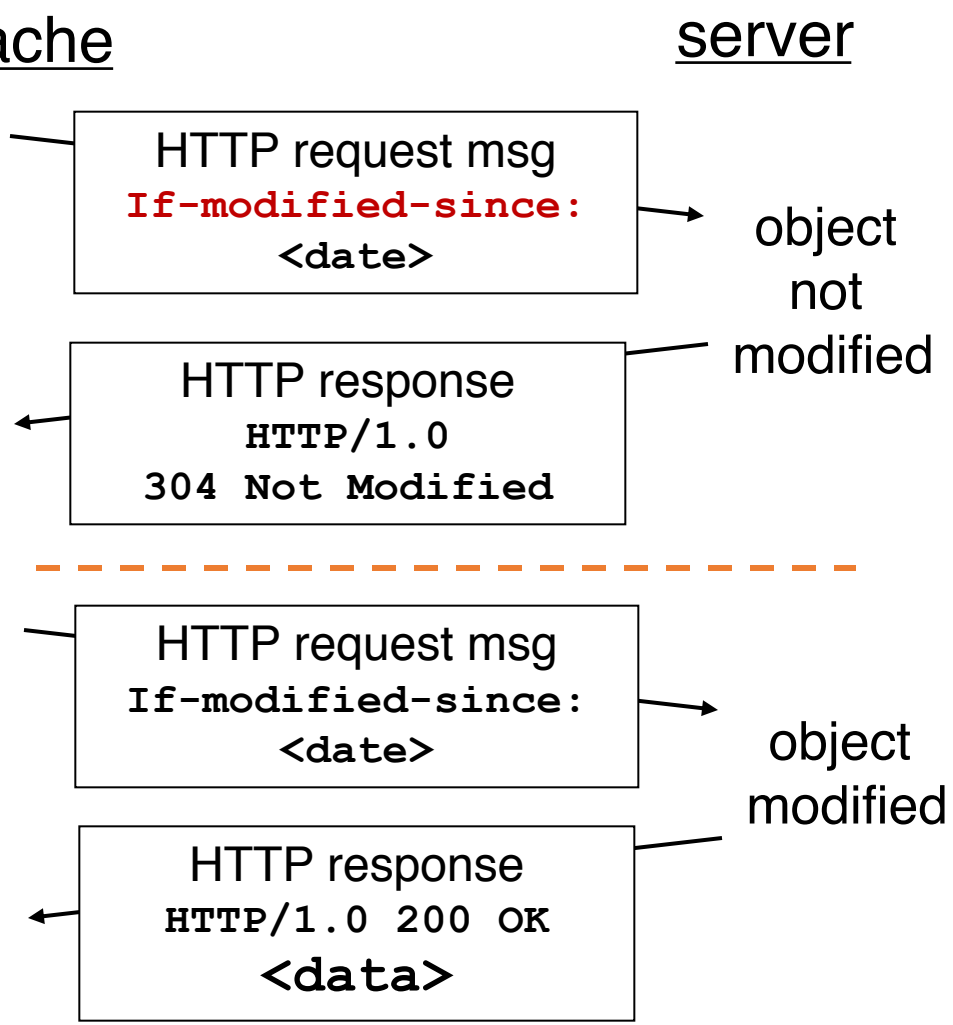


- You can configure a HTTP proxy on your laptop's network settings.
- If you do, your browser sends all HTTP requests to the proxy (cache).
- Hit: cache returns object
- Miss:
  - cache requests object from origin server
  - caches it locally
  - and returns it to client

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

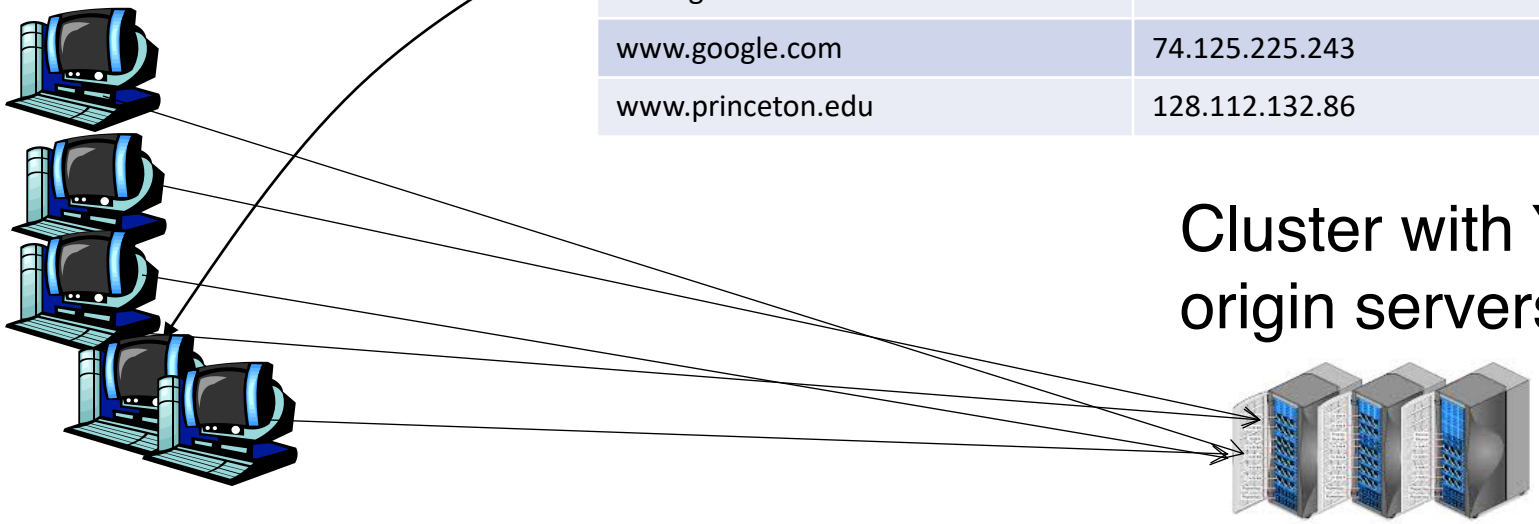
- Provisioned by ISPs
- Or content providers! (Netflix, google, …)

Uses

- Reduce bandwidth requirements of content provider
- Reduce $$ of maintaining servers
- Reduce traffic on the link to the content provider
- Improve response time to user for that 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 Yahoo's origin servers

98.138.253.109

- Huge bandwidth requirements
- 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

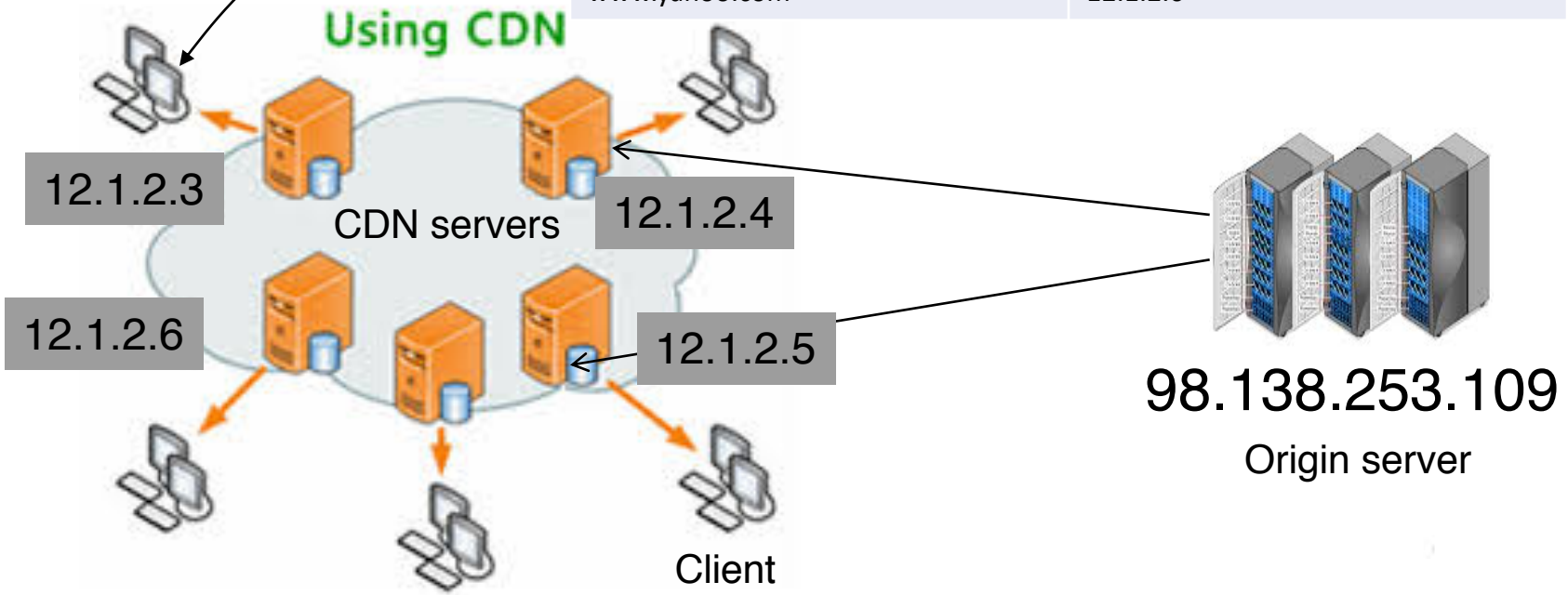
DOMAIN NAME	IP ADDRESS
www.yahoo.com	124.8.9.8 (NS of CDN)
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

Scale through indirection to CDN name server.

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!



Themes from HTTP

- Request/response nature of protocols
  - Headers determine the actions of all the parties of the protocol
- ASCII-based message structures
- Higher performance using caching
- Scaling using indirection
- These principles form of the basis of the web that we enjoy today!