HyperText Transfer Protocol (HTTP)

Computer Networks
Term A15

HTTP Outline

- Web and HTTP Overview
- HTTP (Non-persistent and Persistent)
- HTTP Request and Response Messages
- HTTP/2
- Cookies
- Web Caching with Proxy Servers
- Caching Example

Web and HTTP

Web terminology:

- A web page consists of objects.
- Object can be HTML file, JPEG image, Java applet, audio file,video clip, …
- A web page consists of a base HTML-file which includes several referenced objects.
- Each object is addressable by a URL.
- Example URL:

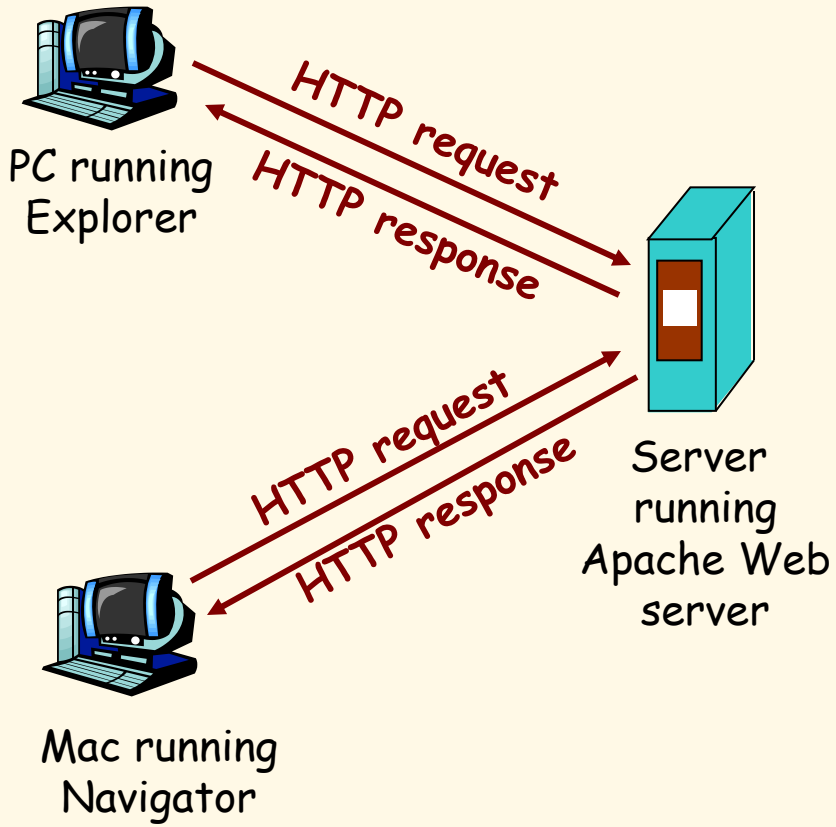
`www.someschool.edu/someDept/pic.gif`

host name | path name

HTTP Overview

HTTP: HyperText Transfer Protocol

- Web's application layer protocol
- client/server model
  - *client:* a browser that requests, receives and "displays" Web objects.
  - *server:* a Web server sends objects in response to requests.



HTTP Overview (continued)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80.
- server accepts TCP connection from client.
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server).
- TCP connection closed.

HTTP is "stateless"

- server maintains no information about past client requests.

aside

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.

HTTP Connections

Non-persistent HTTP

- At most one object is sent over a TCP connection.

Persistent HTTP

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

Nonpersistent HTTP

(contains text, references to 10 jpeg images)

Suppose user enters URL

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

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

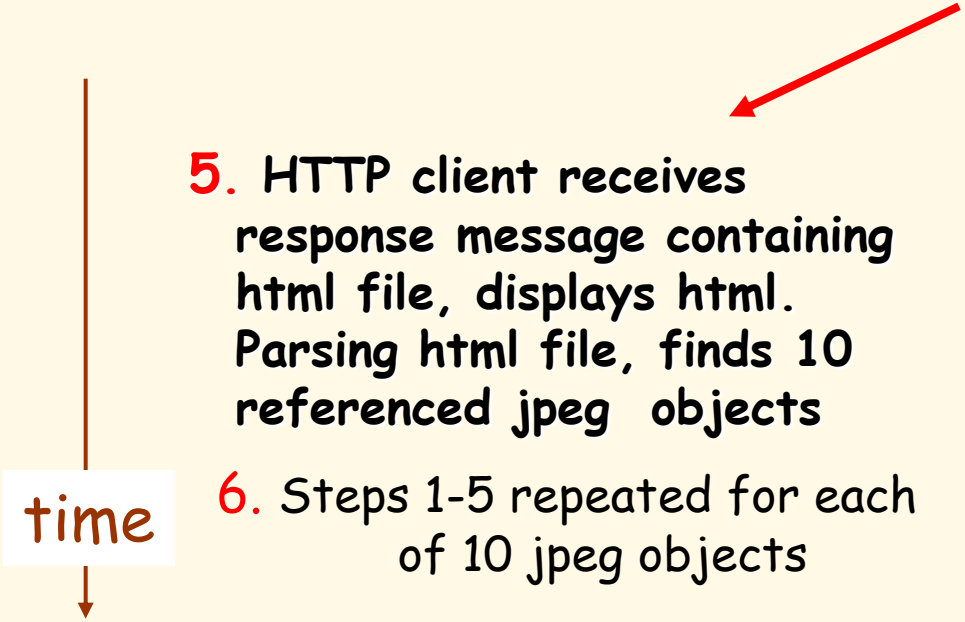
1b. HTTP server at host `www.someSchool.edu` waiting for TCP connection at port 80. "accepts" connection, notifying client.

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

Nonpersistent HTTP (cont.)

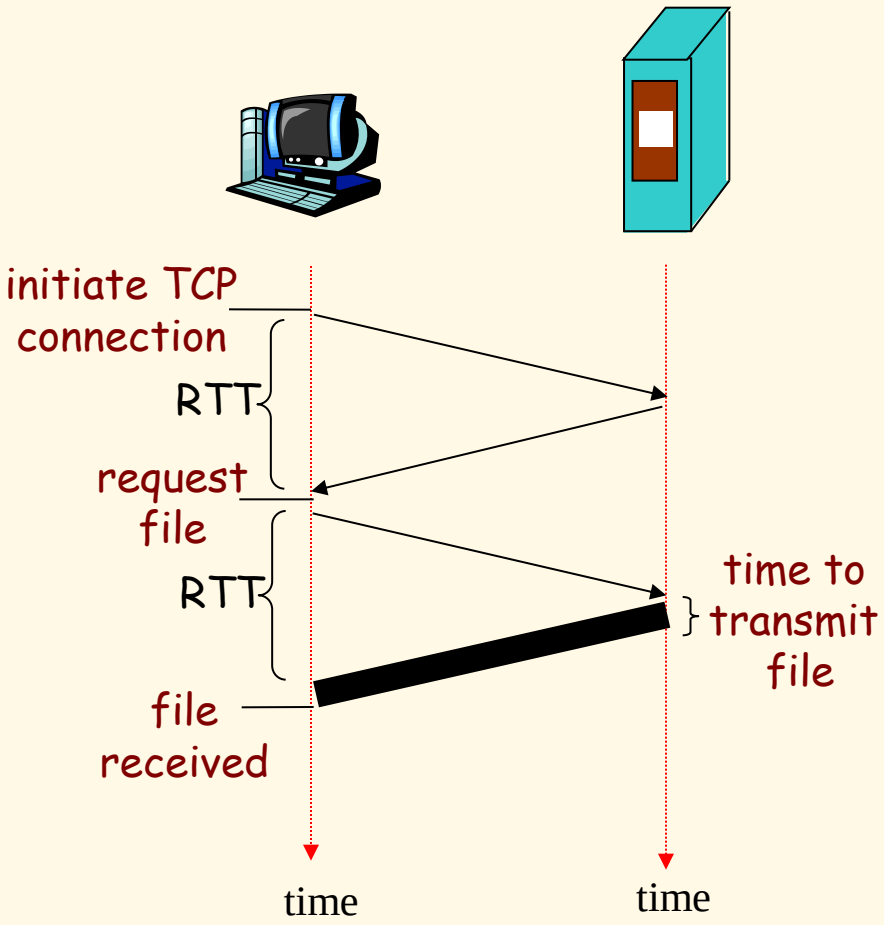


Nonpersistent HTTP: Response Time

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

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 time = 2RTT+transmit time

Persistent HTTP

Nonpersistent HTTP issues:

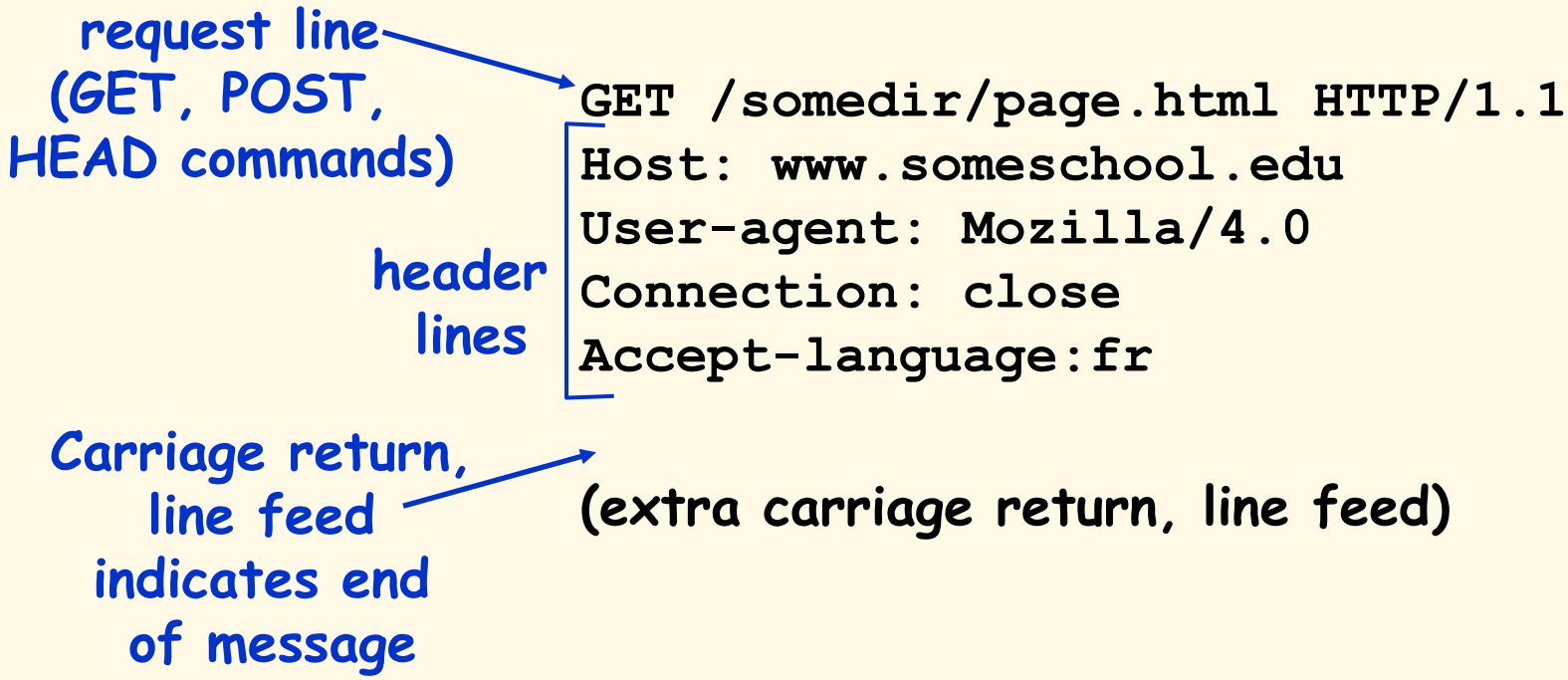
- requires 2 RTTs per object.
- OS overhead for *each* TCP connection.
- browsers often open parallel TCP connections to fetch referenced objects.

Persistent HTTP

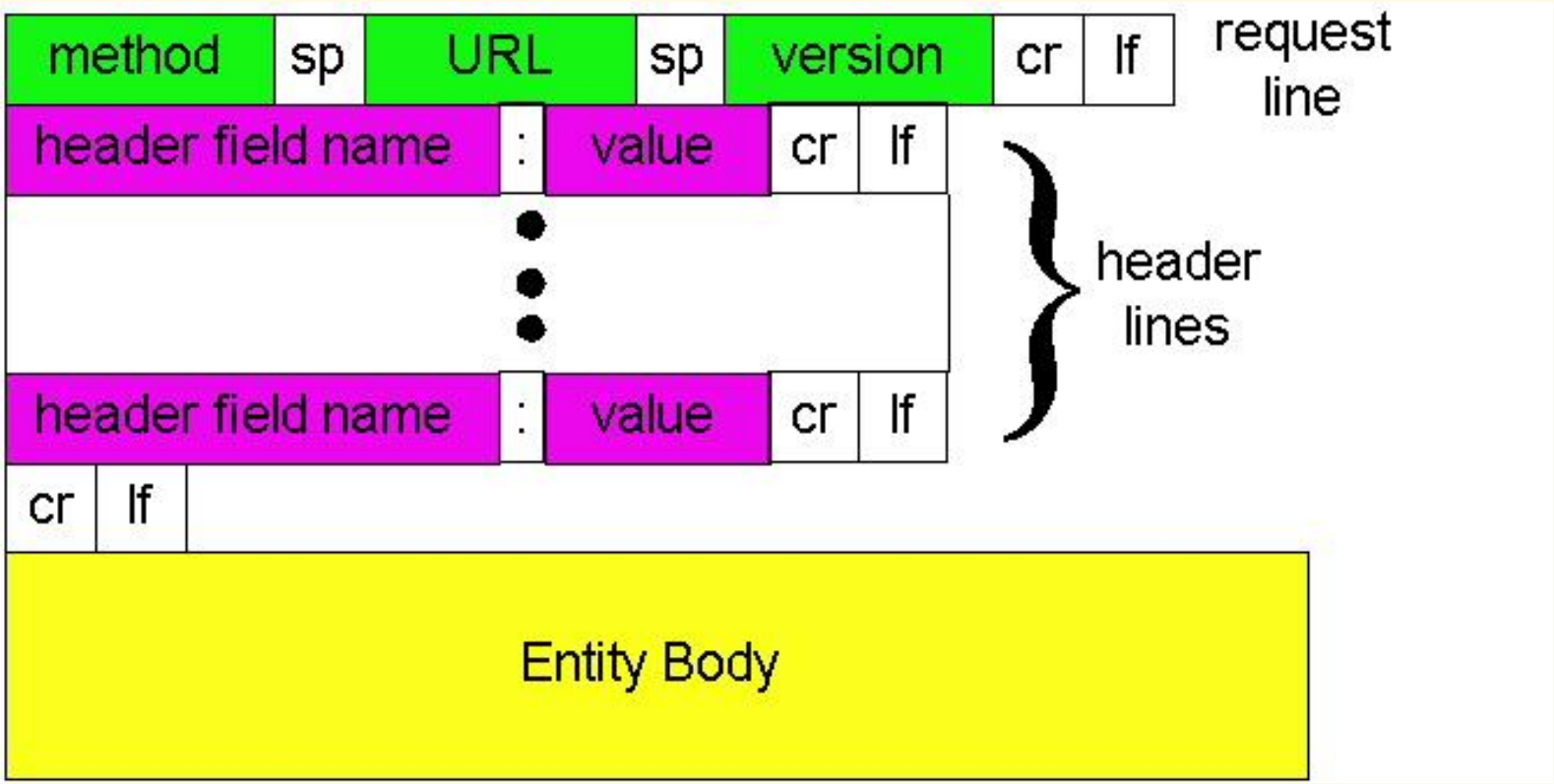
- server leaves connection open after sending response.
- subsequent HTTP messages between same client/server sent over open connection.
- client sends requests as soon as it encounters a referenced object.
- as little as one RTT for all the referenced objects

HTTP Request Message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)



HTTP Request Message: General Format



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
```

Method Types

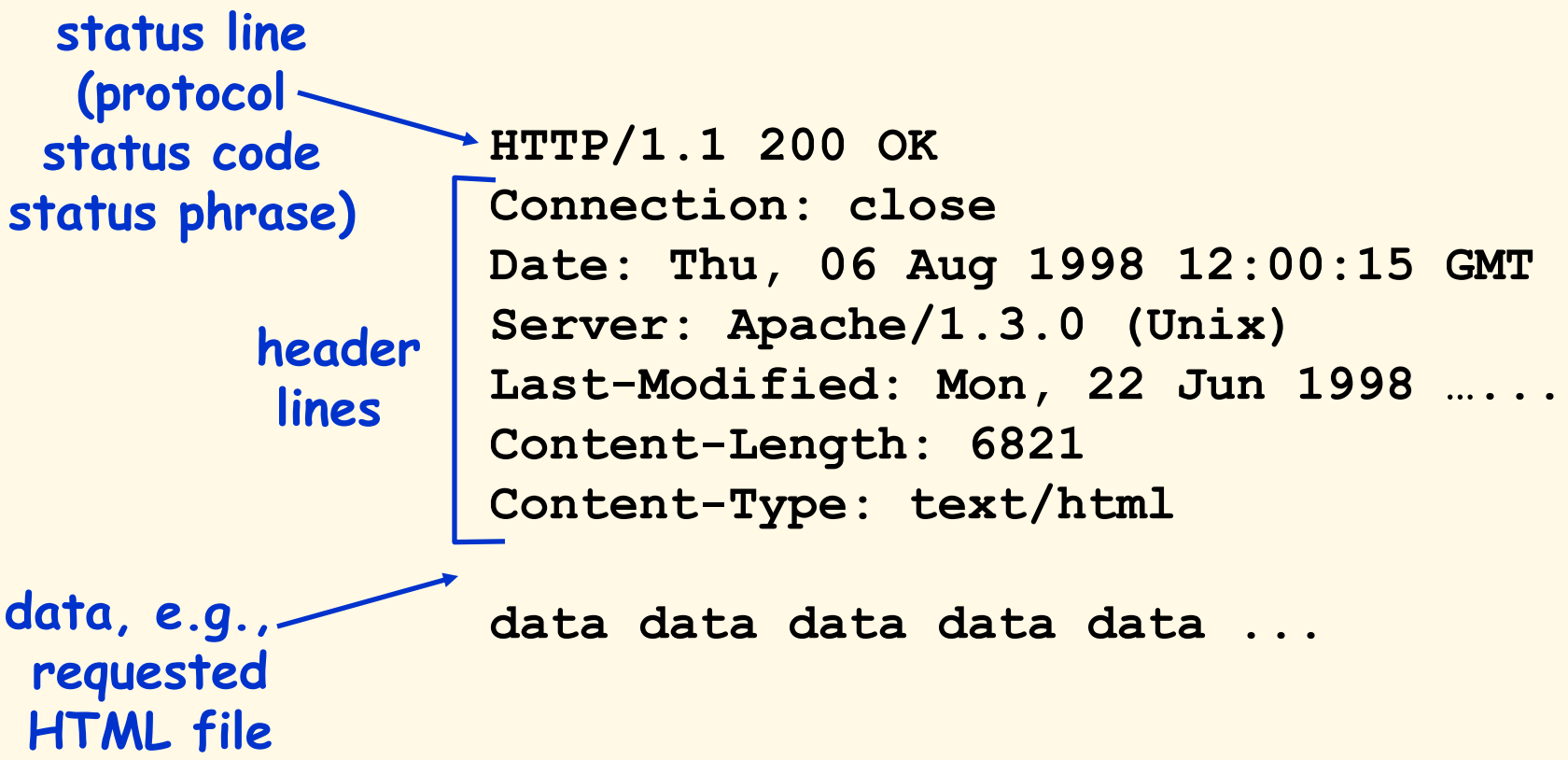
HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

HTTP 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

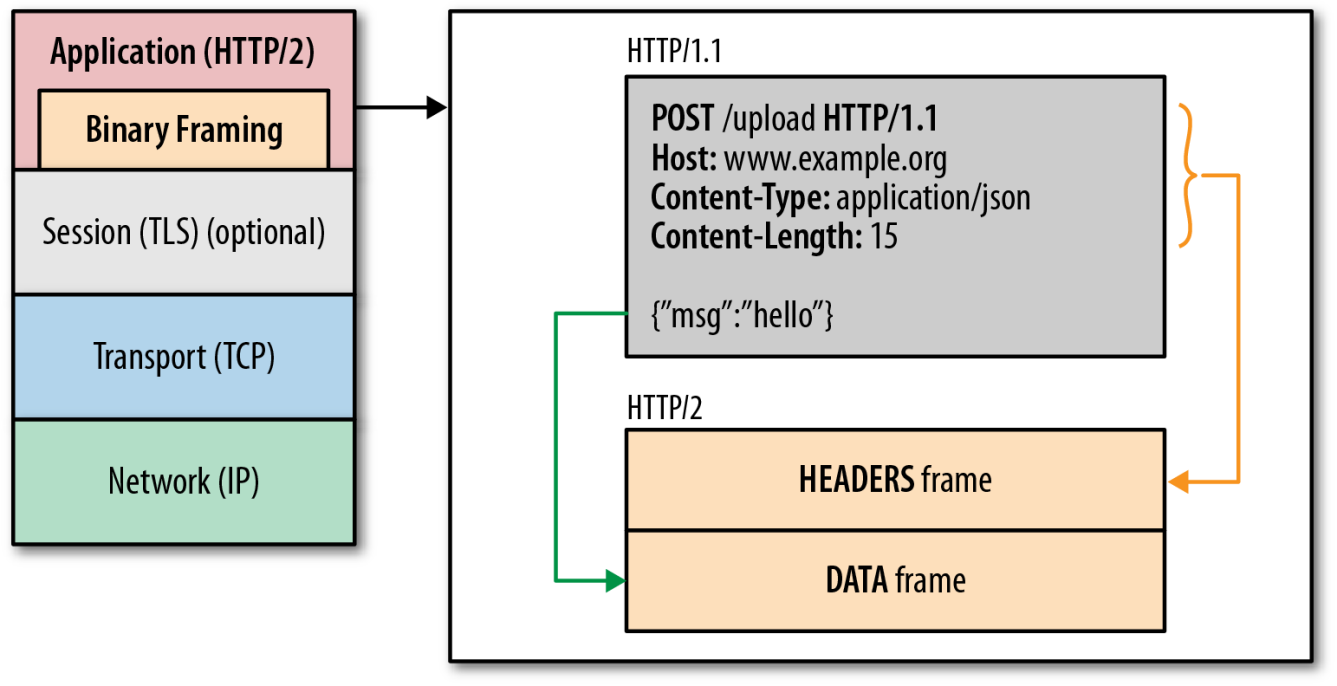
- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

HTTP/2



O'Reilly.com

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet cis.poly.edu 80
```

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

2. Type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.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!

User-server State: Cookies

Many major Web sites use cookies

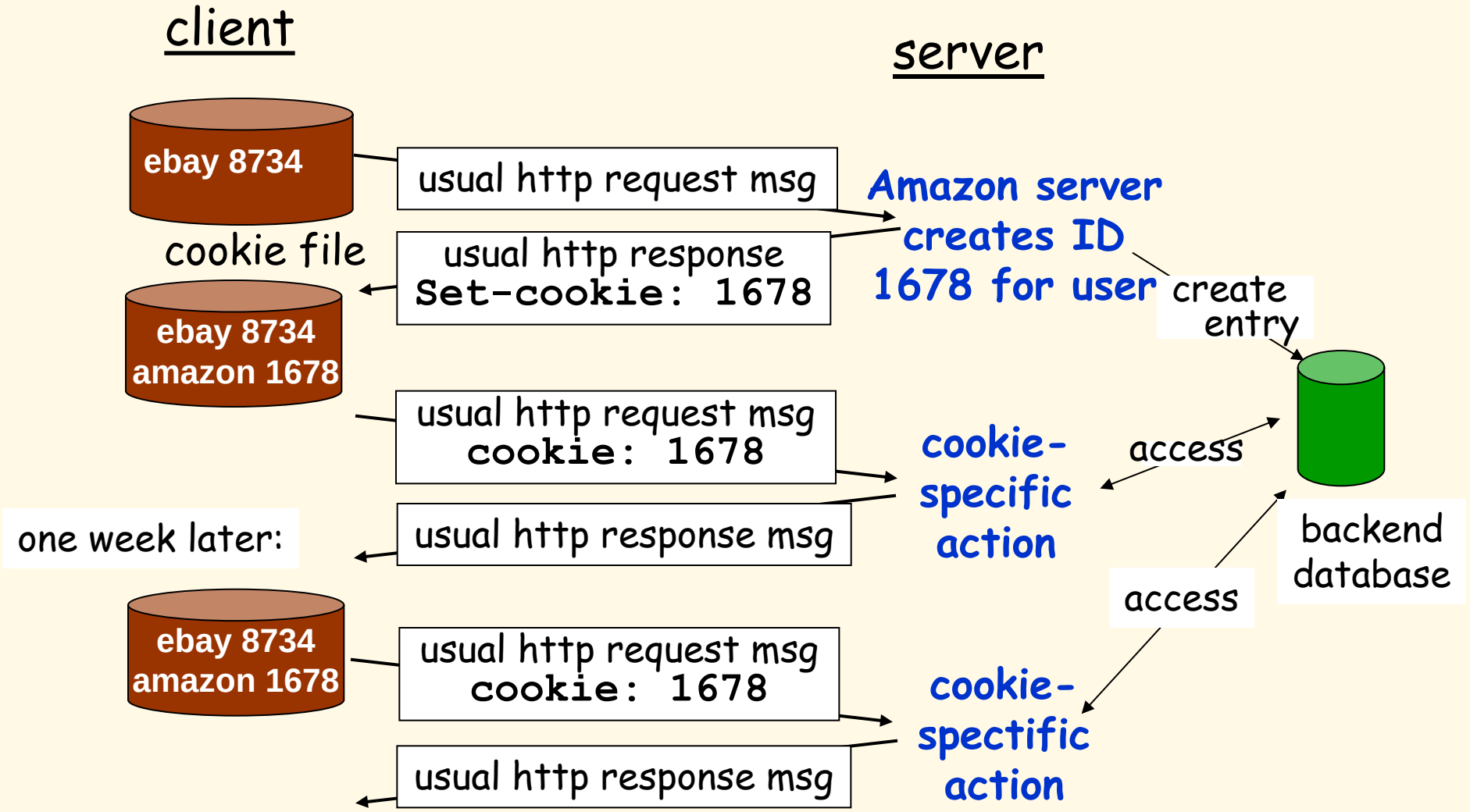
Four components:

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

Example:

- Susan always accesses Internet from PC
- visits specific e-commerce site for first time (Amazon)
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

Cookies: Keeping State



Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

aside

Cookies and privacy:

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

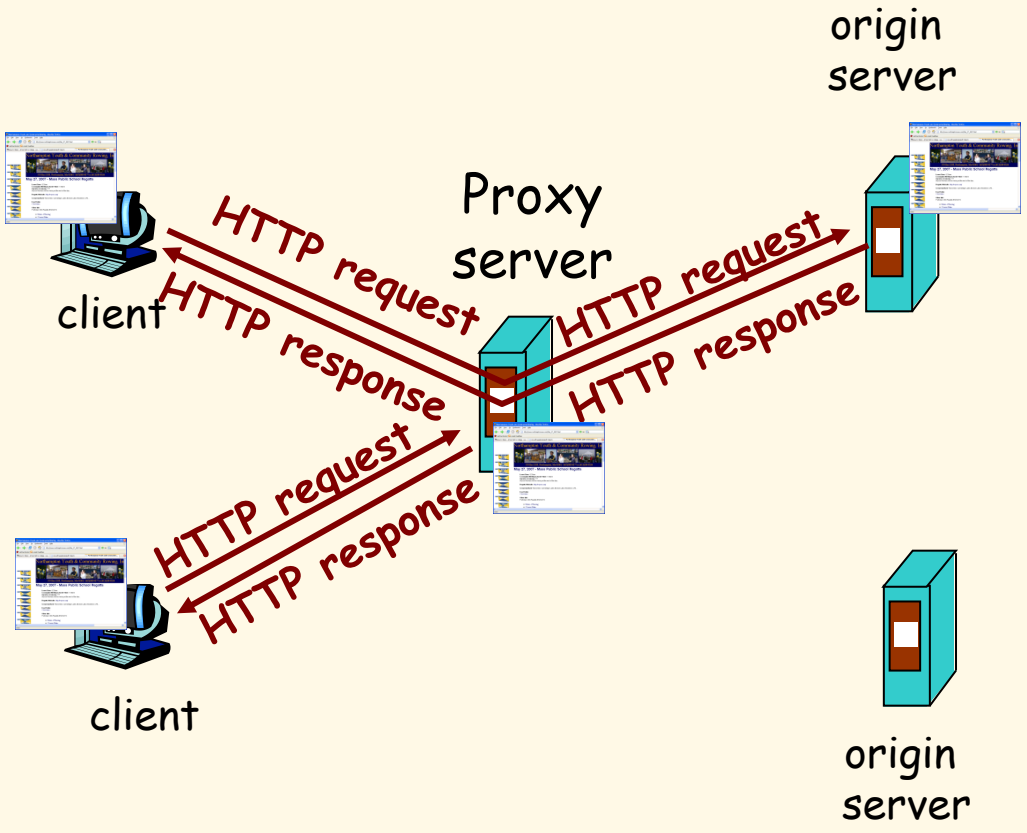
How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies:: http messages carry state.

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

- Cache acts as both client and server
- Typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

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

Caching Example

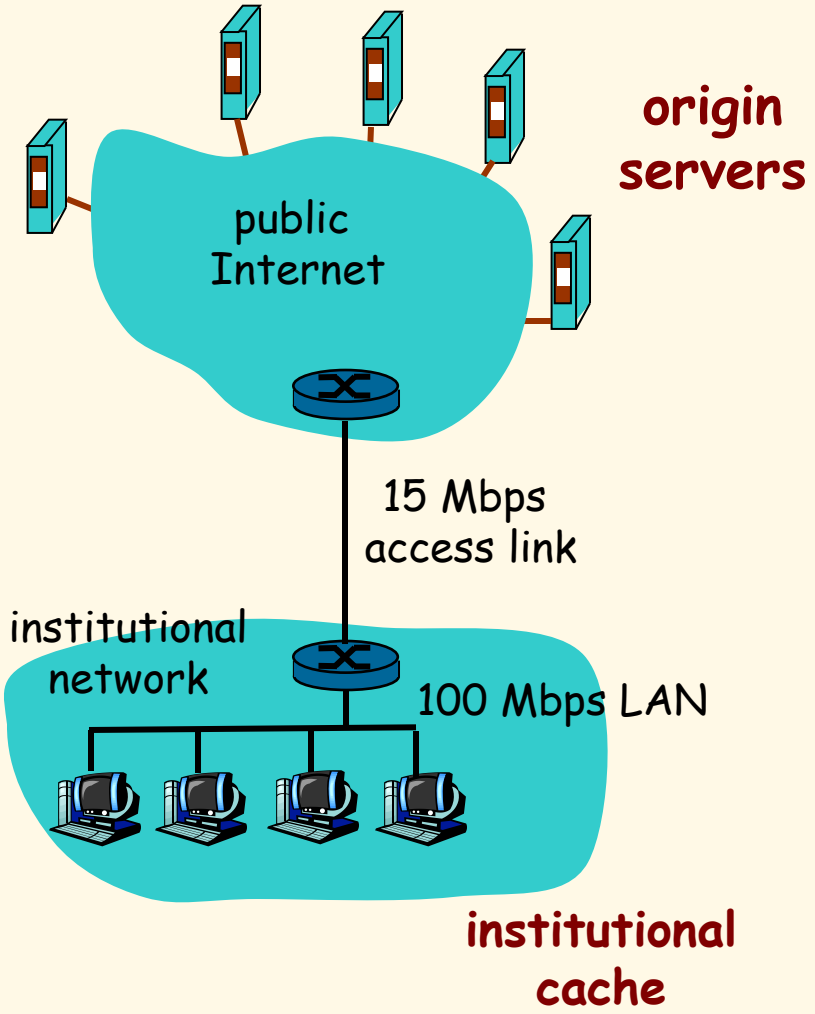
Assumptions

- average object size = 1,000,000 bits
- avg. request rate from institution's browsers to origin servers = 15 requests/sec
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

  = 2 sec + minutes (congested) + milliseconds



Caching Example (cont)

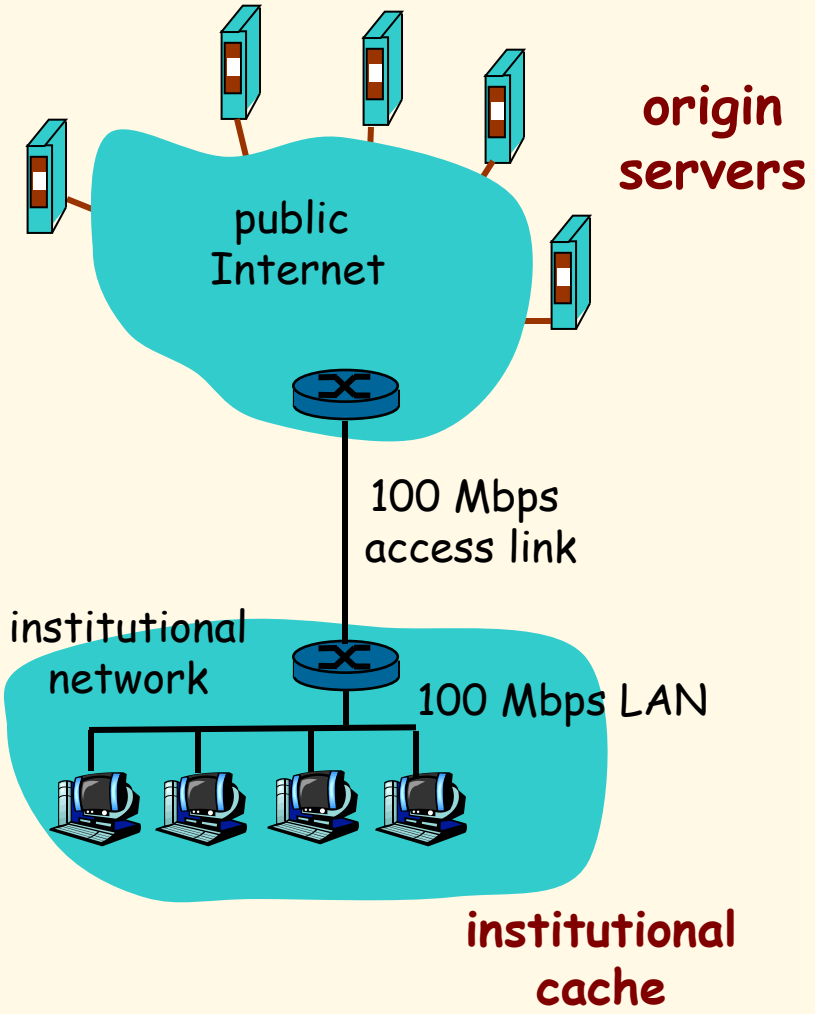
Possible Solution

- increase bandwidth of access link to, say, 100 Mbps

Consequences

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs
- BUT…often a costly upgrade



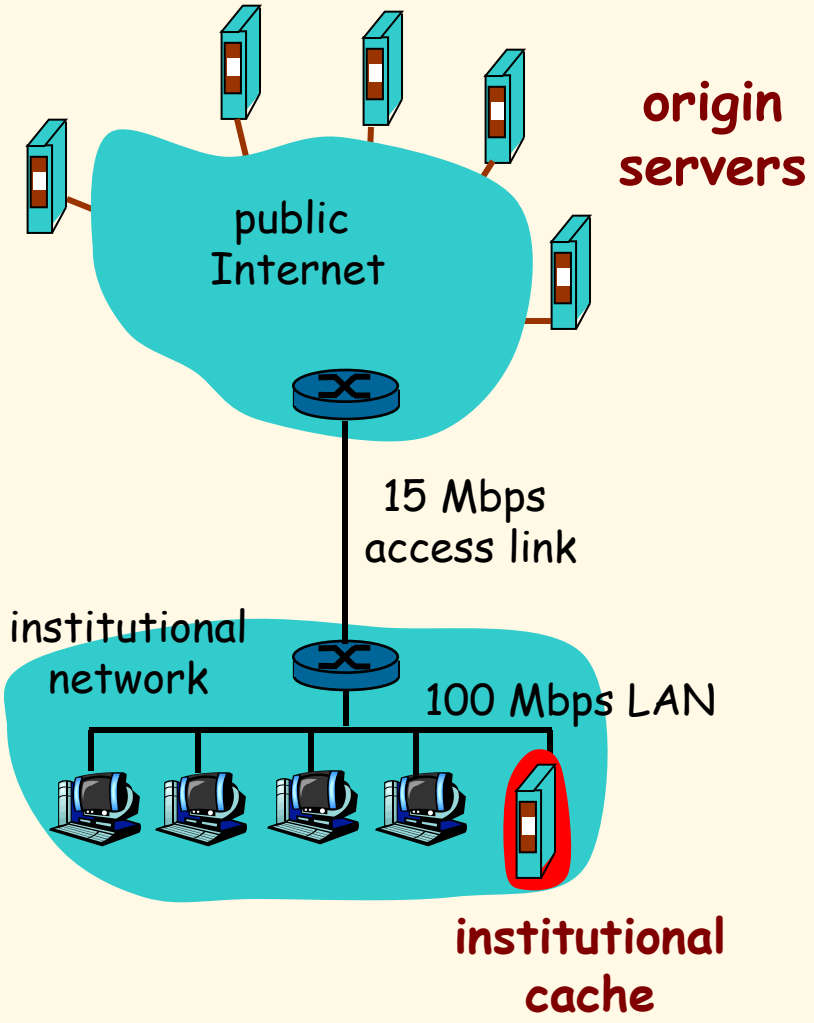
Caching Example (cont)

Possible Solution: Install Cache

- suppose hit rate is 0.4

Consequences

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + .4*milliseconds < 1.4 secs



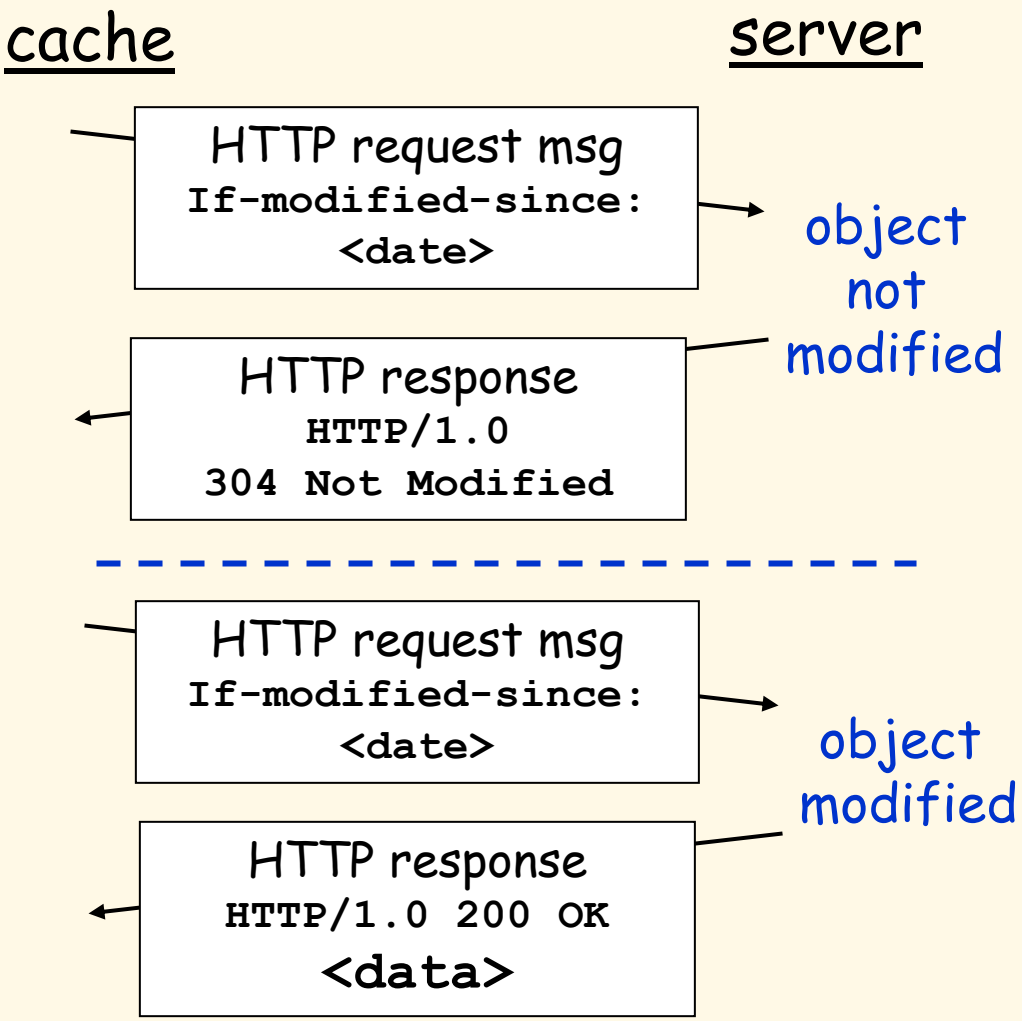
Caching - Conditional GET

- **Goal:** don't send object if cache has up-to-date cached version.
- 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 Summary

- HTTP (Nonpersistent and Persistent)
- HTTP Request and Response Messages
- HTTP/2
- Cookies
- Web Caching with Proxy Servers
- Caching Example