UIC UNIVERSITY OF ILLINOIS AT CHICAGO Department of Computer Science
COLLEGE OF ENGINEERING

CS450 – Introduction to Networking
Lecture 5 – HTTP

Phu Phung

January 23, 2015

Live demo with HTTP (client side) using telnet

1. Telnet to your favorite Web server:

```
telnet www.cs.uic.edu 80
```

opens TCP connection to port 80 (default HTTP server port) at www.cs.uic.edu. anything typed in sent to port 80 at www.cs.uic.edu

2. type in a GET HTTP request:

```
GET /~phu/cs450.html HTTP/1.0
Host: www.cs.uic.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!

Q: To save data from a HTTP response, you should save from:

```
PhuMAC:~ phu$ telnet www.cs.uic.edu 80
Trying 131.193.32.29...
Connected to www.cs.uic.edu.
Escape character is '^]'.
A → GET /~phu/cs450.html HTTP/1.0
    Host: www.cs.uic.edu
B →
    HTTP/1.1 200 OK
C → Date: Fri, 23 Jan 2015 05:31:17 GMT
    Server: Apache/2.2.3 (Red Hat)
    Last-Modified: Tue, 20 Jan 2015 17:05:17 GMT
    ETag: "6f60b27-121-716e8d40"
    Accept-Ranges: bytes
    Content-Length: 289
    Content-Type: text/html; charset=UTF-8
    Connection: close
D →
E → <HTML>
    <BODY>
      <h1>Welcome! </h1>
      This is just a test page for HTTP protocol.
    </BODY>
    </HTML>
    Connection closed by foreign host.
```

Live demo with wireshark

possible structure of applications:

1. Install wireshark: $sudo apt-get install (wireshark link on the course's homepage)
2. Run: $sudo wireshark
    1. Select the connected network interface (In VirtualBox, configure Bridge Adaptor)
3. Open a web browser and browse a URL
4. Examine the captured HTTP request/ response in wireshark

HTTP overview

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 sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

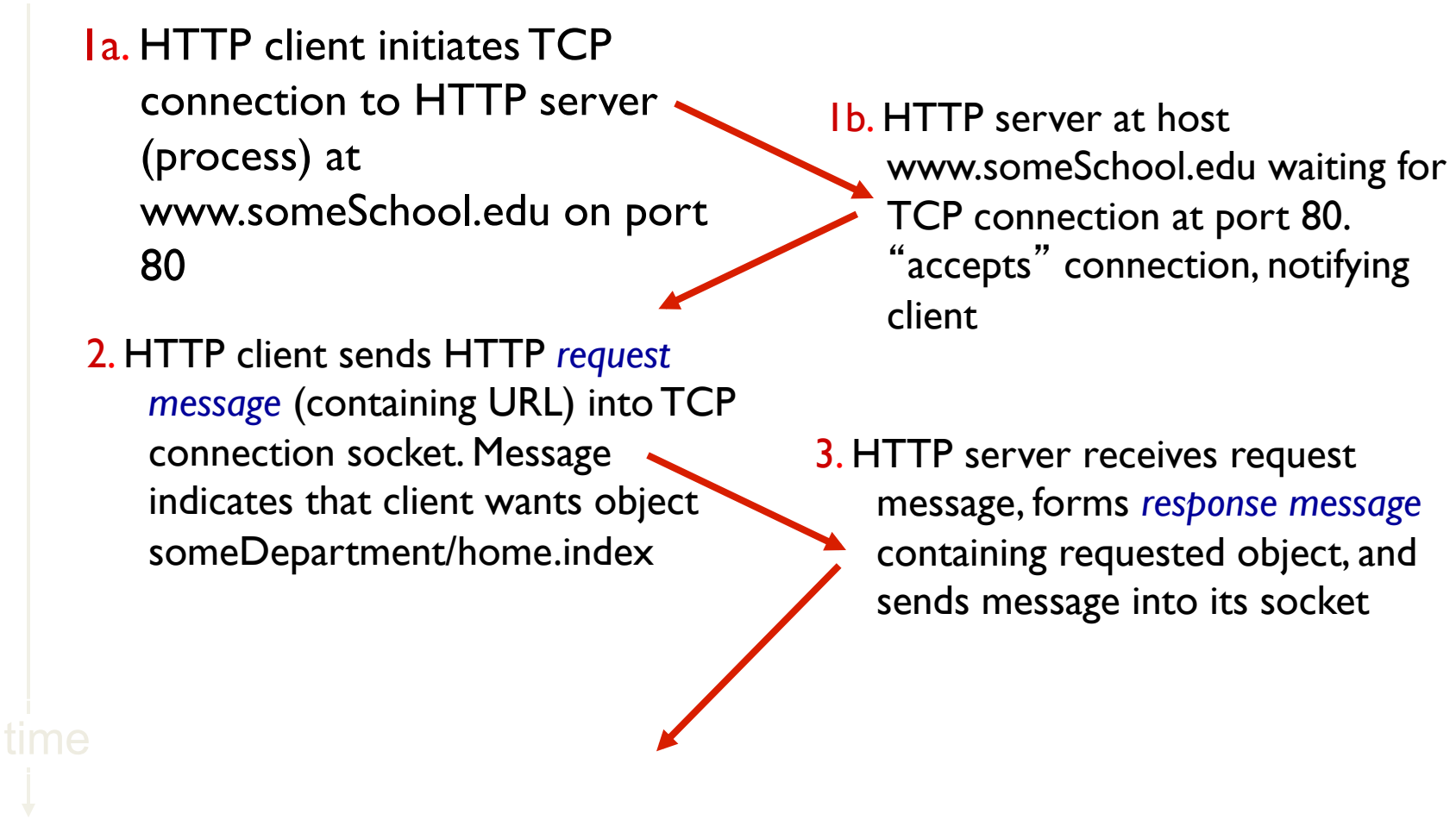
persistent HTTP

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

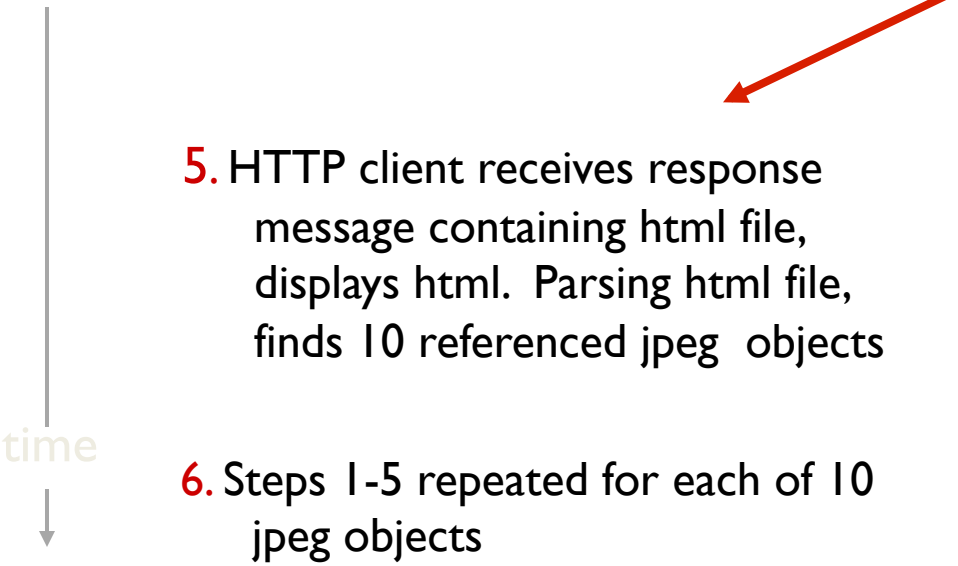
Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



Non-persistent HTTP (cont.)



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

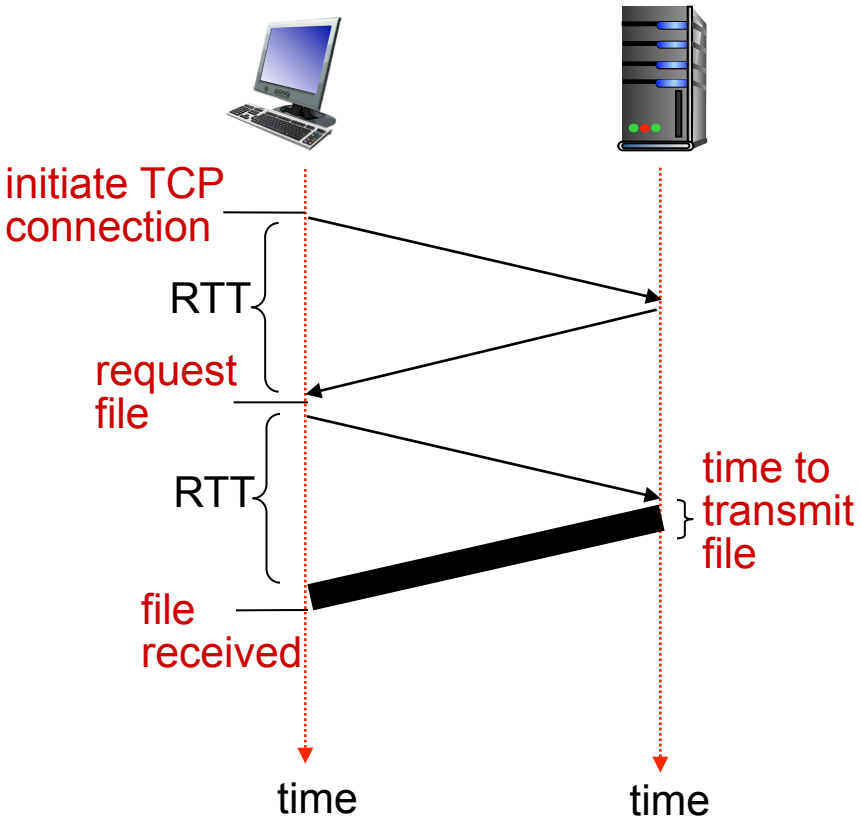
Non-persistent HTTP: response time

RTT (definition): 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 =

  2RTT+ file transmission time



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

Select a correct statement

A. Persistent HTTP uses more connections.

B. Persistent HTTP sends fewer HTTP requests

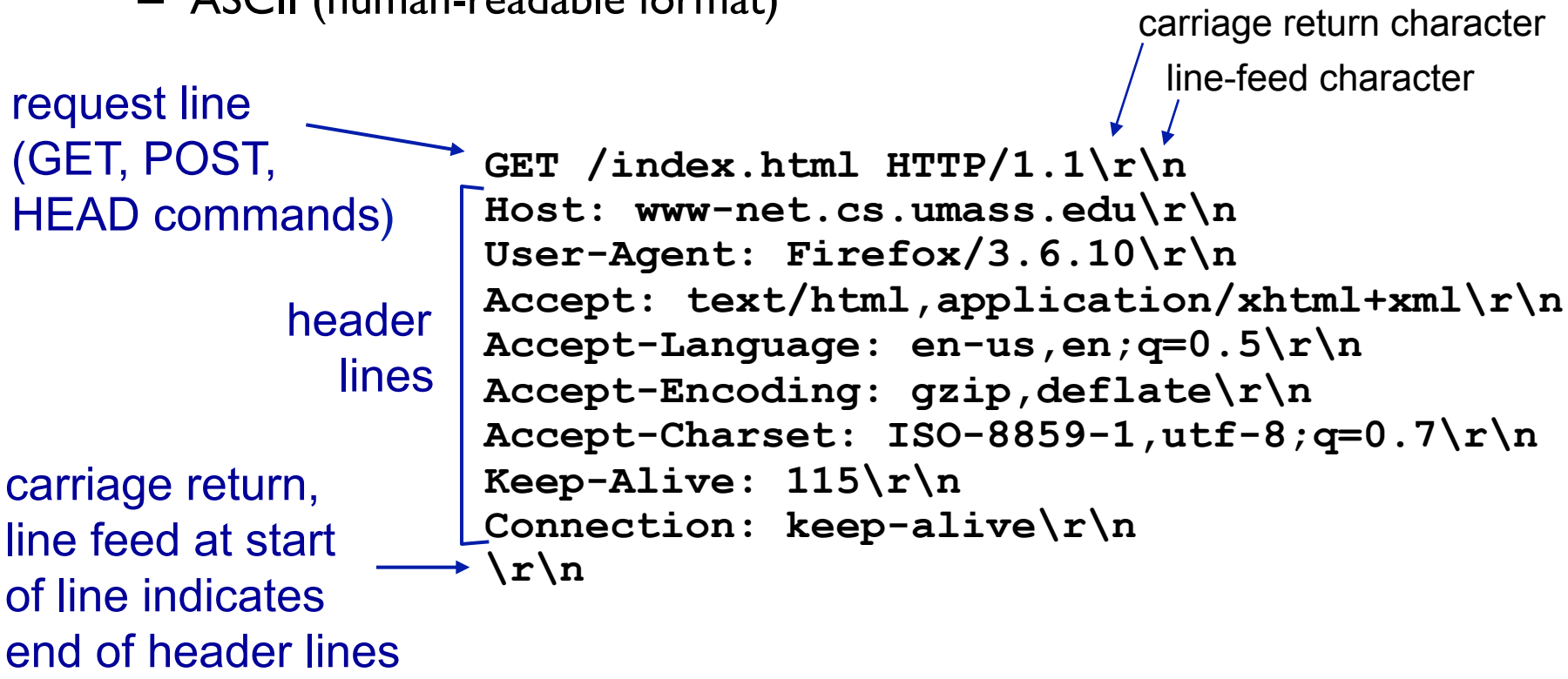
C. Persistent HTTP sends more HTTP requests

D. Persistent HTTP uses less connections to improve download time

E. Both B and D are correct

HTTP request message

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



Why \r\n (CRLF) is used in HTTP request?

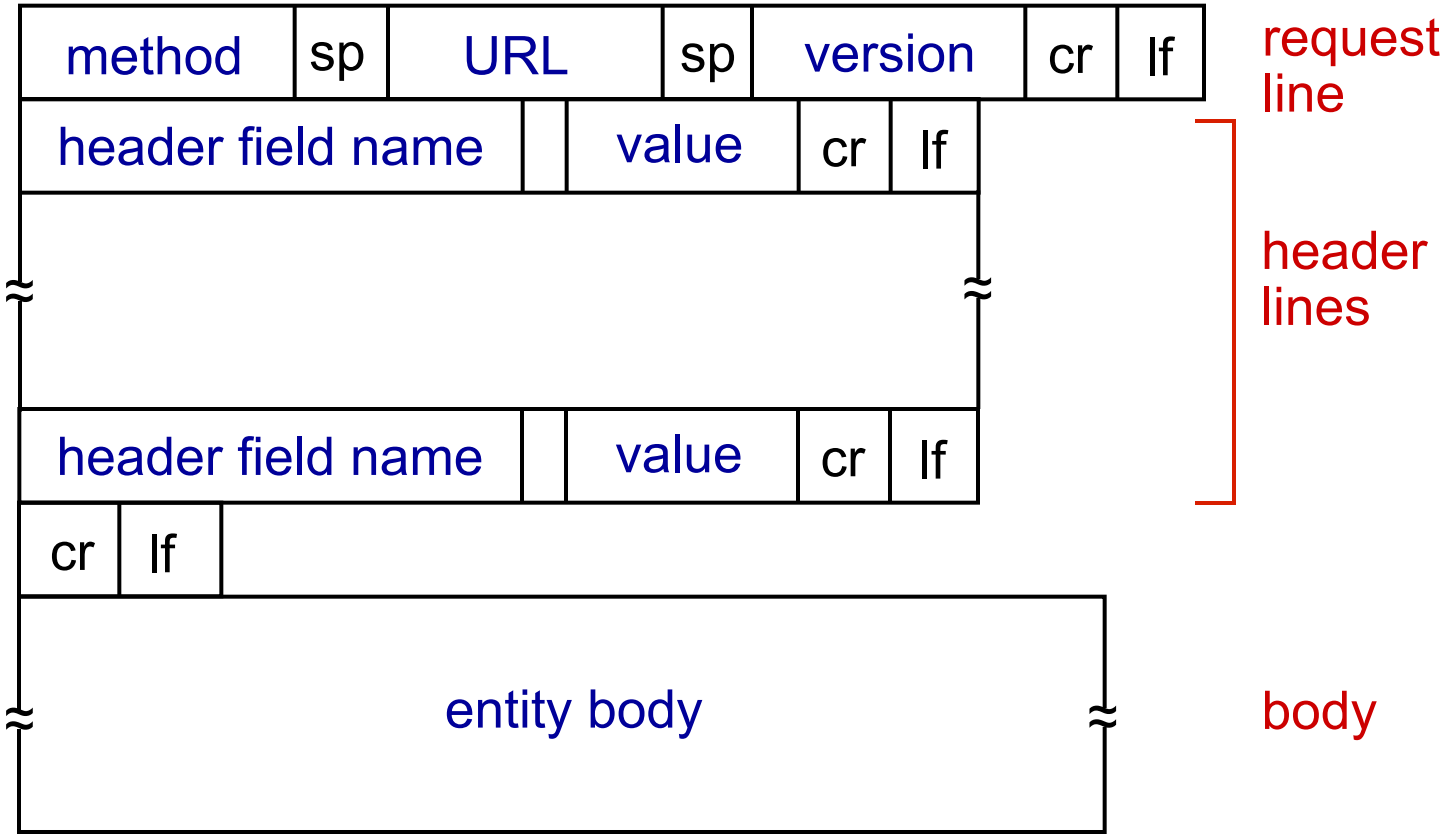
A. Just unnecessary protocol format

B. They are automatically generated when user hits ENTER

C. They are used to indicate the end of a header field or section

D. None of above is correct

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

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: Keep-Alive\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.
- some sample codes:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

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

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Different features between HTTP 1.0 vs 1.1

- Extensibility
- Caching
- Bandwidth optimization
- Network connection management
- Message transmission
- Internet address conservation
- Error notification
- Security, integrity, and authentication
- Content negotiation

Reference: http://www8.org/w8-papers/5c-protocols/key/key.html

Why cookies are used in HTTP

A. It is required by law

B. To improve non-persistent HTTP

C. Because HTTP is stateless

D. All above are correct

User-server state: cookies

many Web sites use cookies

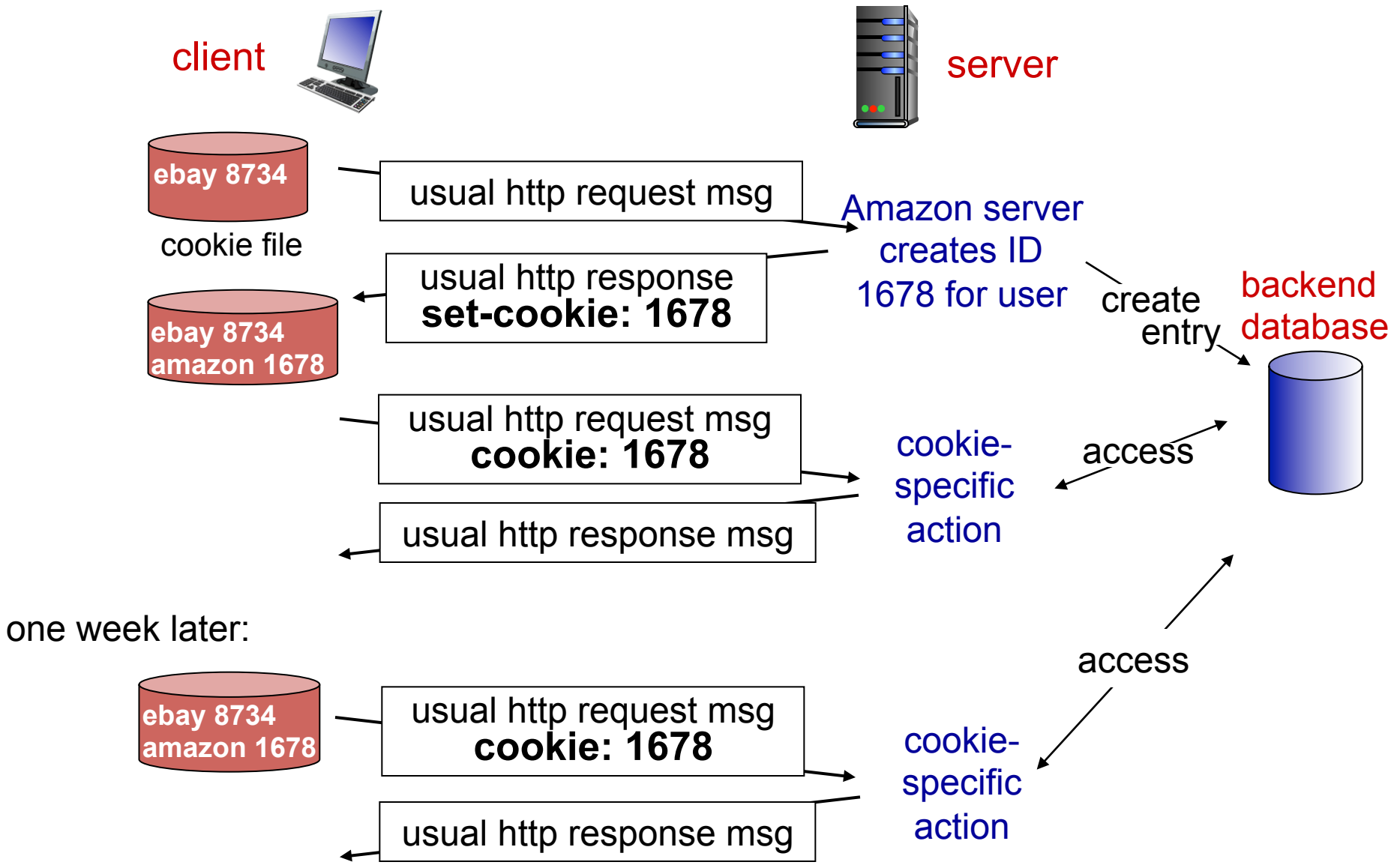
four components:

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 (continued)

what cookies can be used for:

- 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

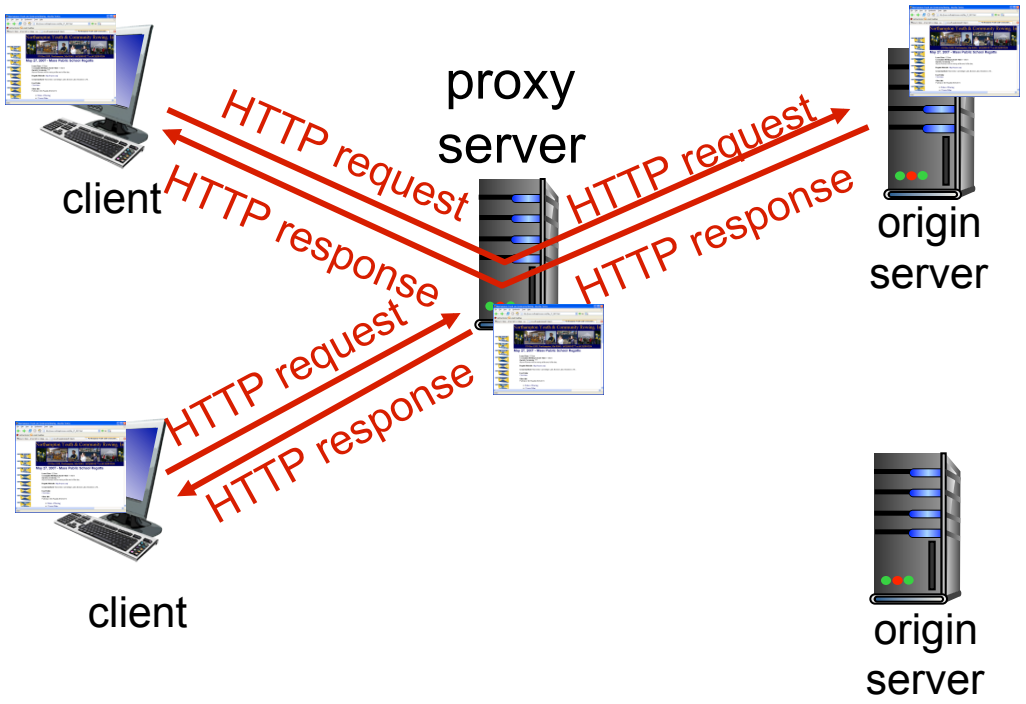
how to keep "state":

- 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
  - server for original requesting client
  - client to origin server
- 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

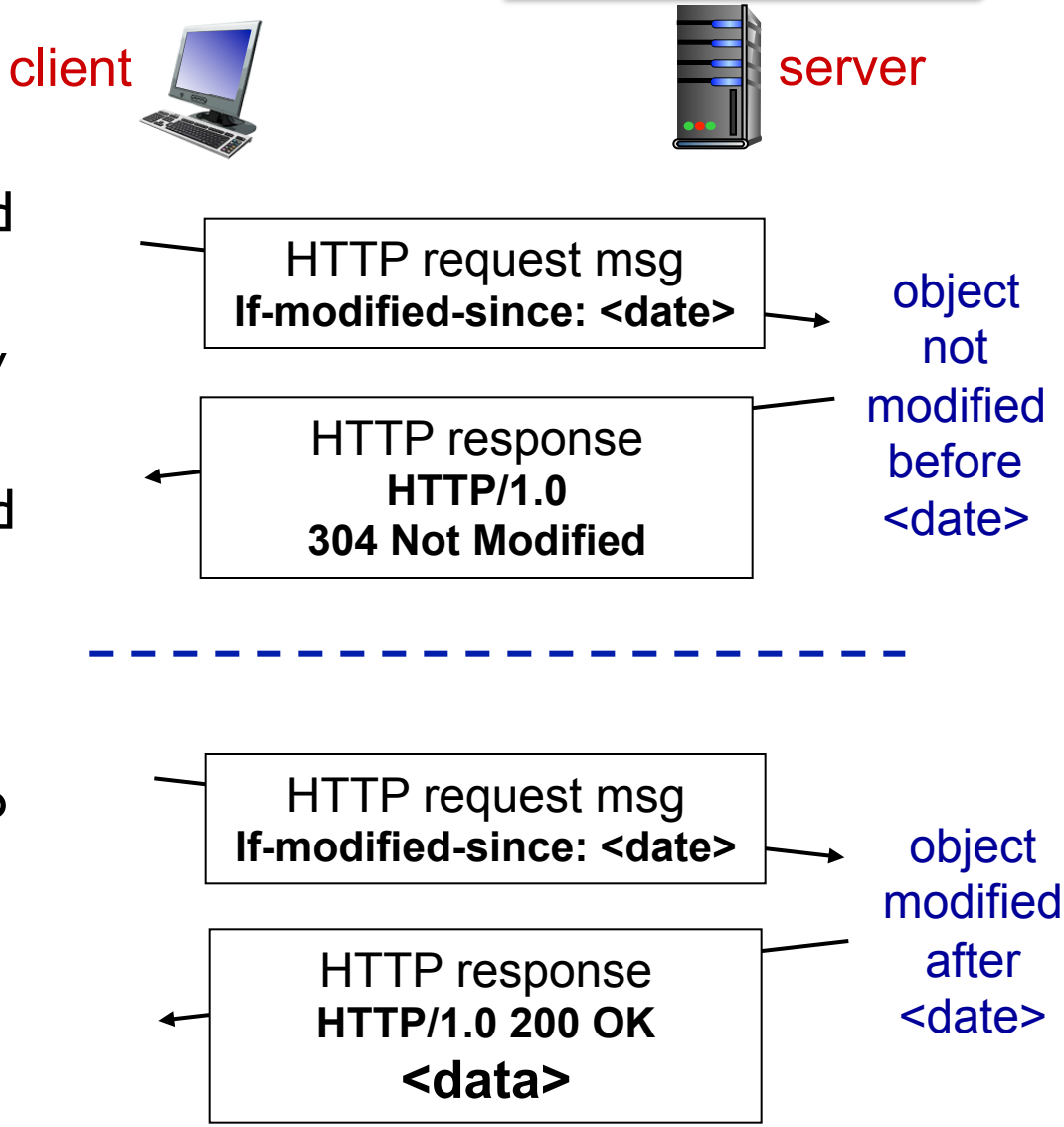
Let's see in wireshark

- *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
```



A quick check on assignment 1 progress

A. I have completed/nearly completed the project

B. I have done about >= 75%

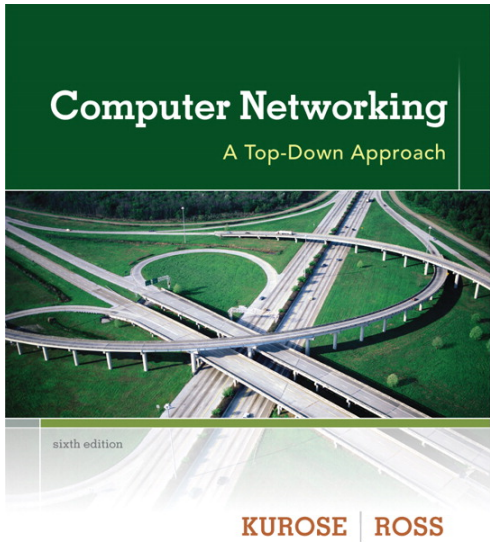
C. I have done about >= 50%

D. I have done about >= 25%

E. I want the deadline extended

Next lecture

- Client-Server Model and Assignment 2
- SMTP
  - Readings 2.4
- Questions regarding Assignment 1?
- Assignment 2 will be posted by next lecture



Copy right notice:
These slides are adapted from J.F Kurose and K.W. Ross's ones

Computer Networking: A Top Down Approach

6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012