CS 372 Lecture #10

The Application Layer:

- Example application
  - Hypertext Transfer Protocol (HTTP)

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Web and HTTP (RFC 2616)

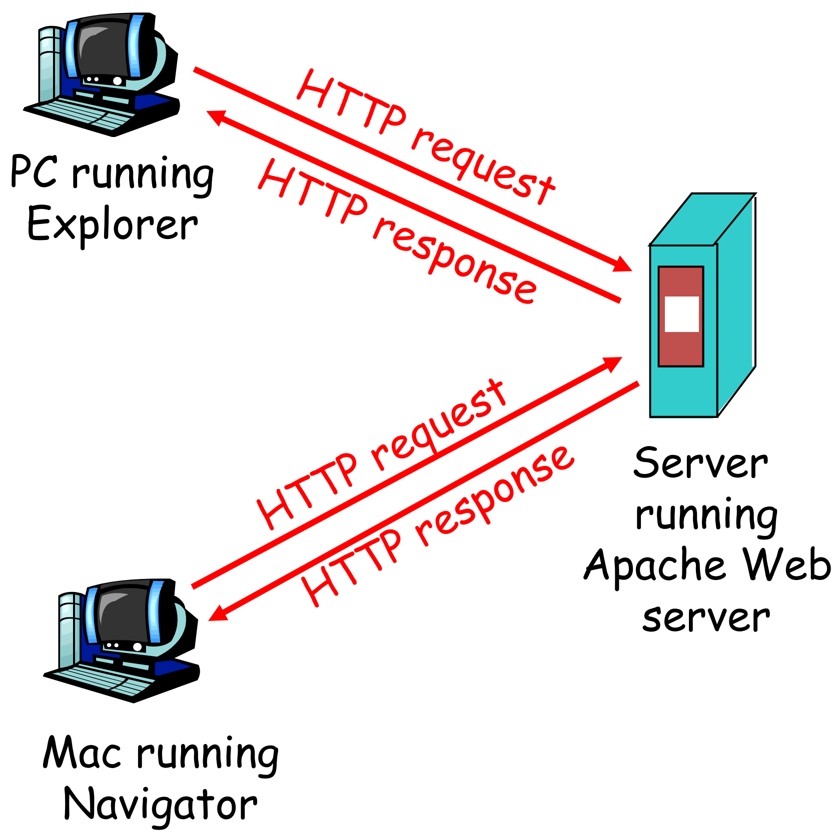
- Web page consists of a base HTML-file which includes several referenced objects
  - Object can be HTML file, JPEG image, Java applet, audio file,…
- Each object is addressable by a Uniform Resource Locator (URL)

- Example URL:

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\substack{\text{path name in host}\\ \text{directory structure}}}$$

HTTP overview

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* 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

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- downloading multiple objects requires multiple connections

Persistent HTTP

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

Non-persistent HTTP

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) on port 80 at `www.someSchool.edu`

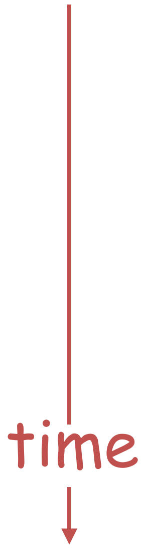
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

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, and closes connection.

time

Non-persistent HTTP (cont.)



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

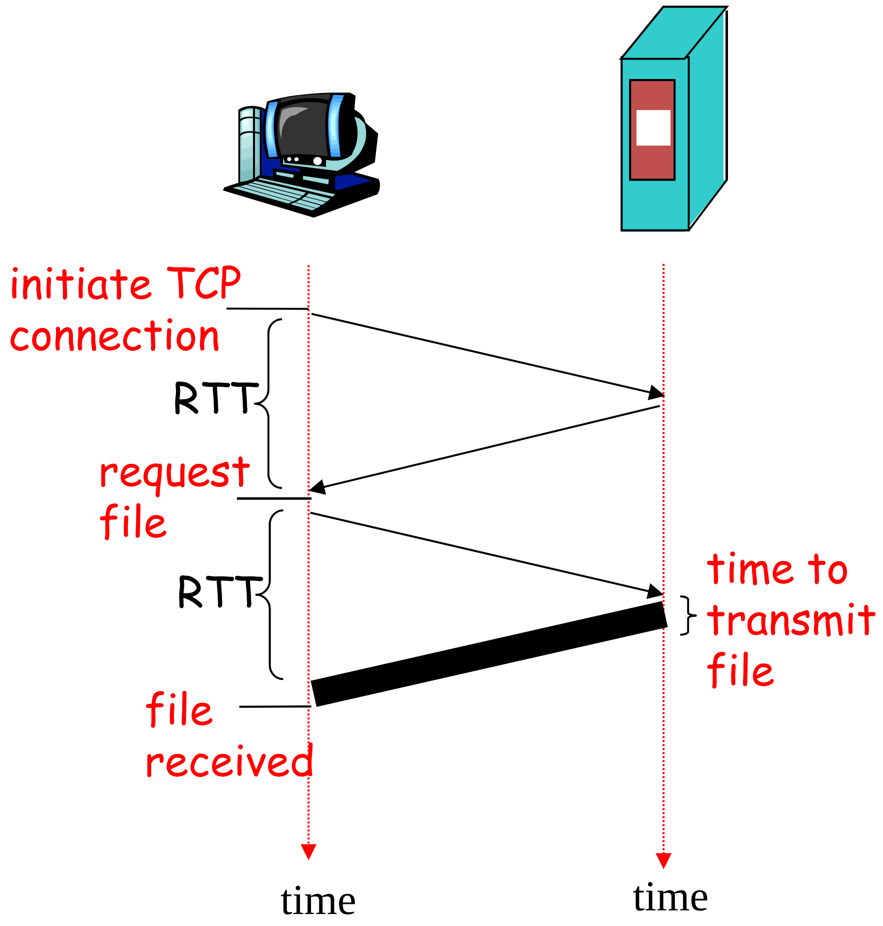
5. Steps 1-4 repeated for each of 10 jpeg objects

Non-Persistent HTTP: Response time

Definition of RTT(round-trip time): 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 = 2RTT+transmit time

(non-persistent HTTP)

Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- operating system 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)

carriage return character

line-feed character

request line
(GET, POST,
HEAD commands)

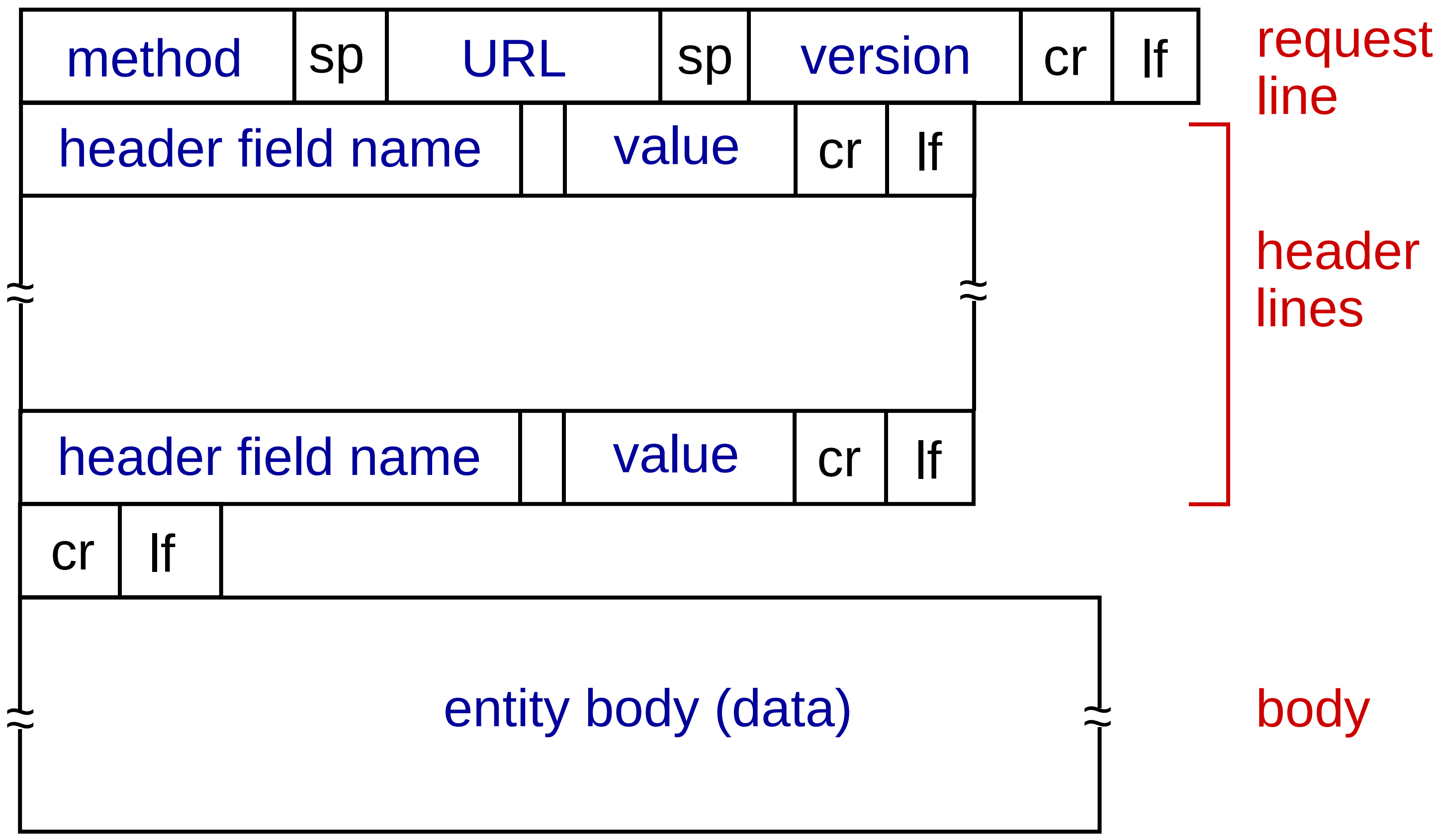
header
lines

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

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Your web browser is an application that includes and implements the client side of HTTP.

HTTP request message: general format



sp = space   cr/lf = carriage-return/line feed (*Enter key*)

Uploading form input

Web page often includes form input

URL method:

- uses GET method
- input is uploaded in URL field of request line
- Example:

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

POST method:

- input is uploaded to server in entity body

HTTP response message

status line
- protocol
- status code
- status phrase

Header lines

Extra cr/lf

data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 24 Mar 2013 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 19 Mar 2013 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

In first line of server-to-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.engr.oregonstate.edu 80
```

Opens TCP connection to port 80
(default HTTP server port) at web.engr.oregonstate.edu
Anything you type will be sent to port 80 at web.engr.oregonstate.edu
and the connection will close (default behavior).

2. Type in a GET HTTP request:

```
GET /~paulson/ HTTP/1.1
Host: engr.oregonstate.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

(or use Wireshark to look at captured HTTP request/response)

Summary Lecture #10

- Definitions
  - URL
  - RTT
  - response time

- HTTP
  - Non-persistent, persistent
  - Request, response
  - Form input