

APPLICATION LAYER
HTTP - FTP - TELNET

Web and HTTP

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

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

protocol

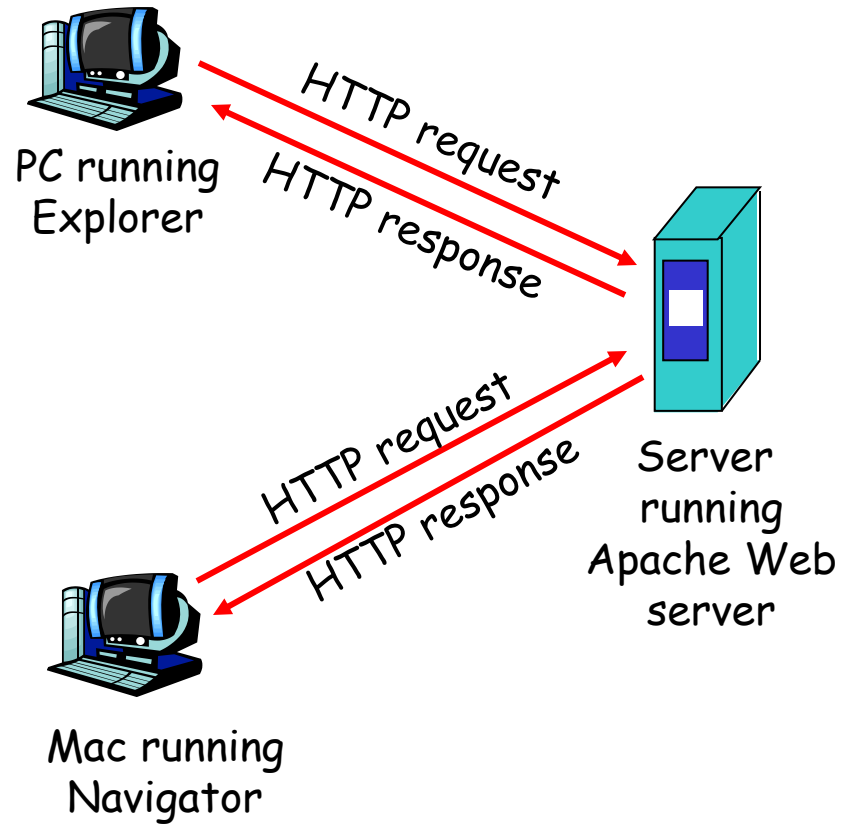
host + network name
("www" is host name)

path (full file) name
(or directory name, or "")

HTTP overview

HTTP: hypertext transfer protocol

- 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 1.0: RFC 1945
- HTTP 1.1: RFC 2068
- **RFC: specification**



HTTP Overview

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80 (default)
- 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

HTTP connection

Nonpersistent HTTP

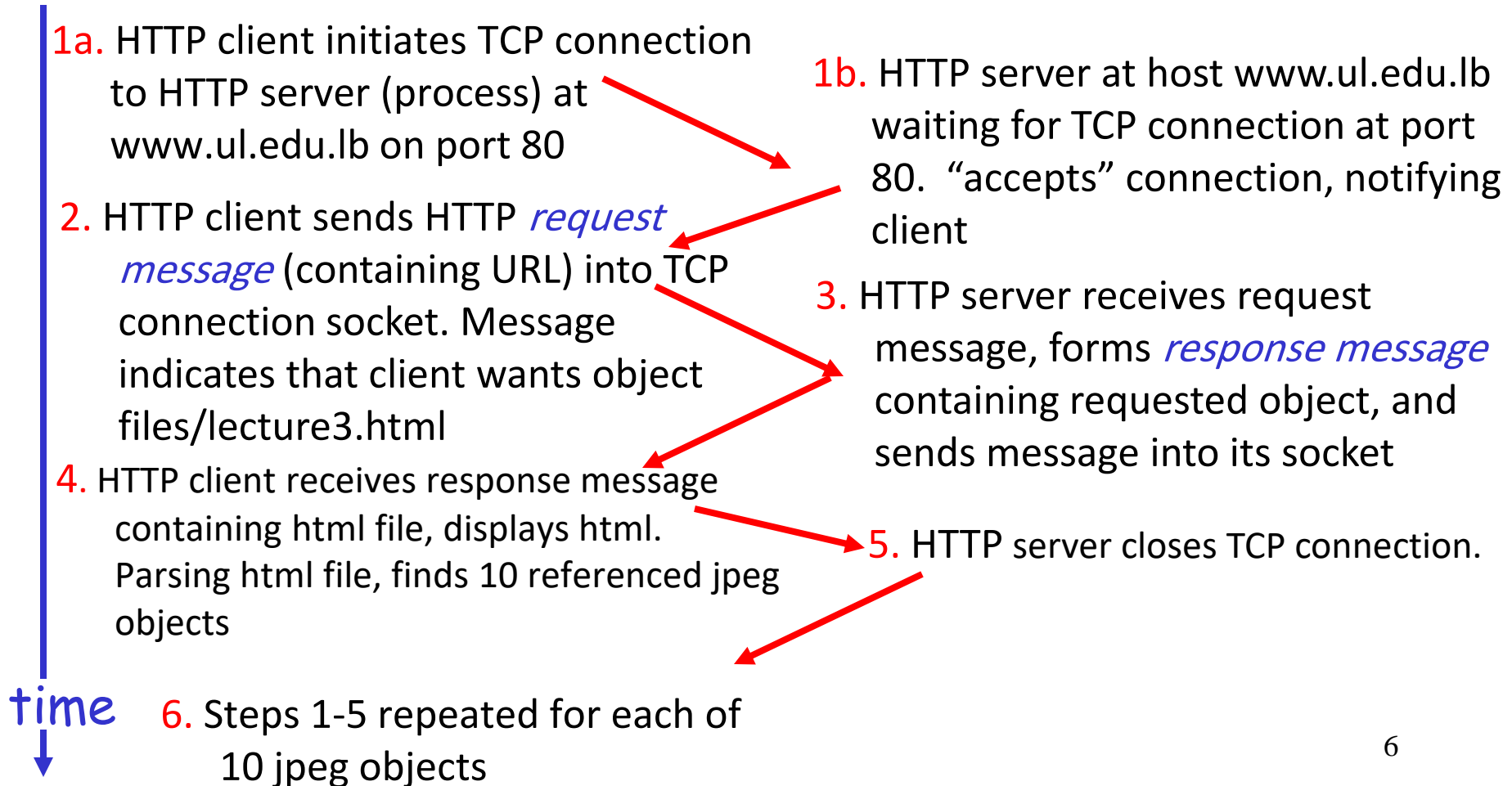
- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

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

Nonpersistent HTTP

- When you enter the following URL to your navigator:
www.ul.edu.lb/files/lecture3.html



Non-persistent HTTP: Response time

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

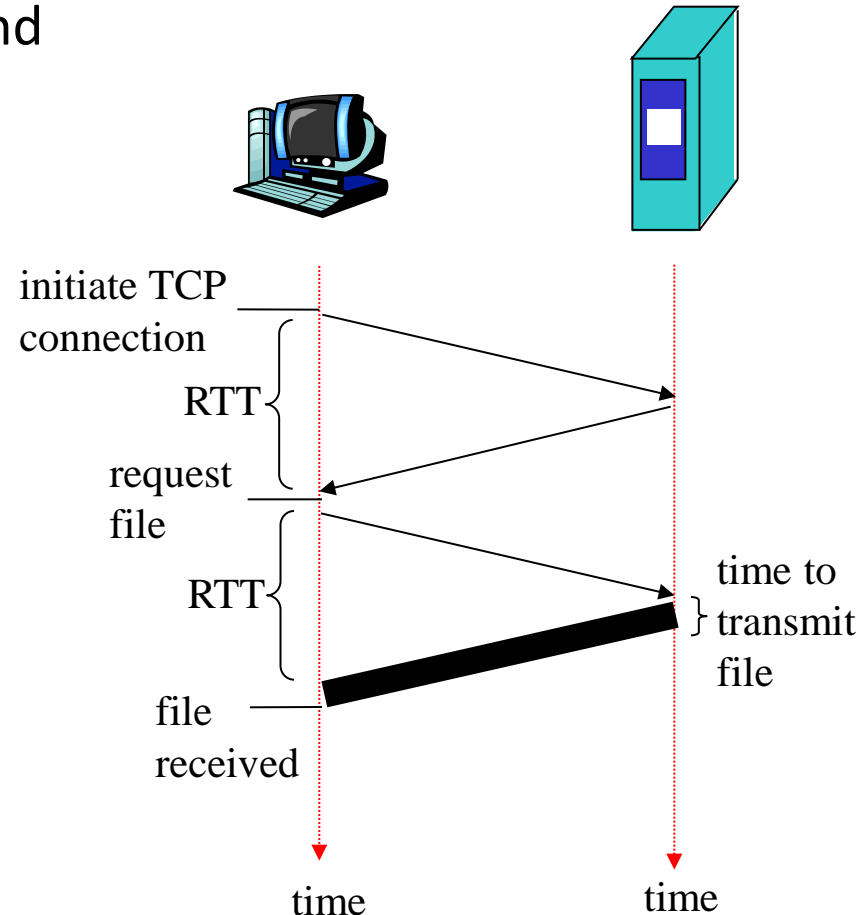
Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = $2RTT + \text{transmit times}$

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

Persistent *without* pipelining:

- client issues new request only when previous response has been received → one RTT for each referenced object

Persistent *with* pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP: method type

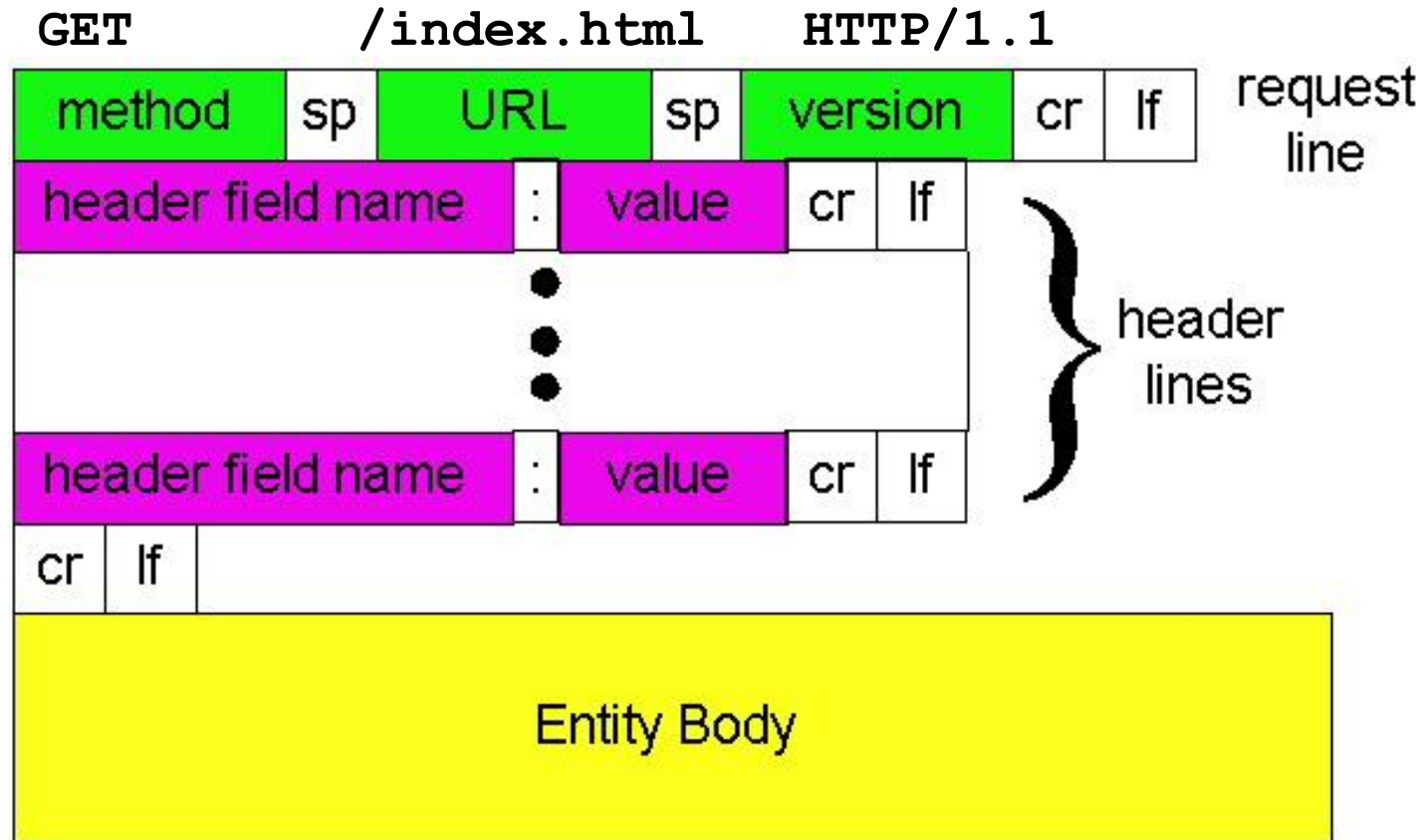
HTTP/1.0

- 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 request message: general form



sp: separator can be a space

cr: carriage return

lf: line feed

Example: HTTP request message

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

request line
(GET, POST,
HEAD commands)

header
lines

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

Carriage return,
line feed
indicates end
of message

(extra carriage return, line feed; i.e., a blank line)

Uploading form input

Post method:

- Web page often includes form input
- Input is uploaded to server in entity body (when you click "Enter")

URL method:

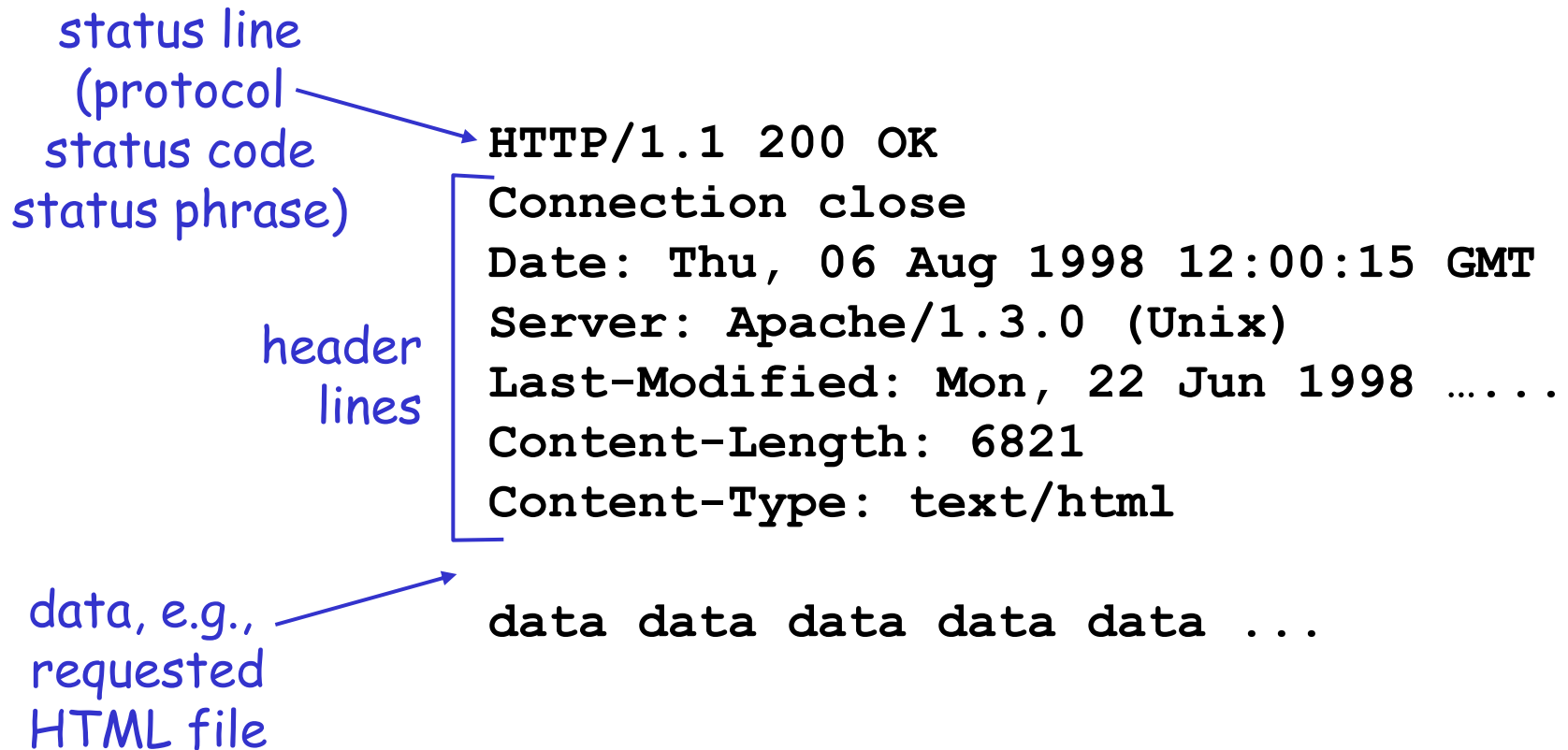
- Uses GET method
- Input is uploaded
- in URL field of request line

www.somesite.com/animalsearch?monkeys&banana=3

? - start of request parameters

& - start of next parameter (name=value)

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

LAB: telnet

Telnet is a tool to connect to a remote computer (telnet /?)

telnet can be used to communicate with a web server.

Type telnet www.ul.edu.lb 80

(80 is used to specify a port number. HTTP uses port 80. By default Telnet uses port 23)

Try the GET command

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 access Internet always from same PC
- She visits a specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Example: cookies

client

server

Cookie file
ebay: 8734

usual http request msg
usual http response +
Set-cookie: 1678

server
creates ID
1678 for user

entry in backend
database

Cookie file
amazon: 1678
ebay: 8734

usual http request msg
cookie: 1678
usual http response msg

cookie-
specific
action



access

access

one week later:

Cookie file
amazon: 1678
ebay: 8734

usual http request msg
cookie: 1678
usual http response msg

cookie-
specific
action

LAB: cookies

Cookies are installed on your computer.

Under windows 7, cookies are in directory:

C:\Users\BH\AppData\Local\Microsoft\Windows\Temporary Internet Files

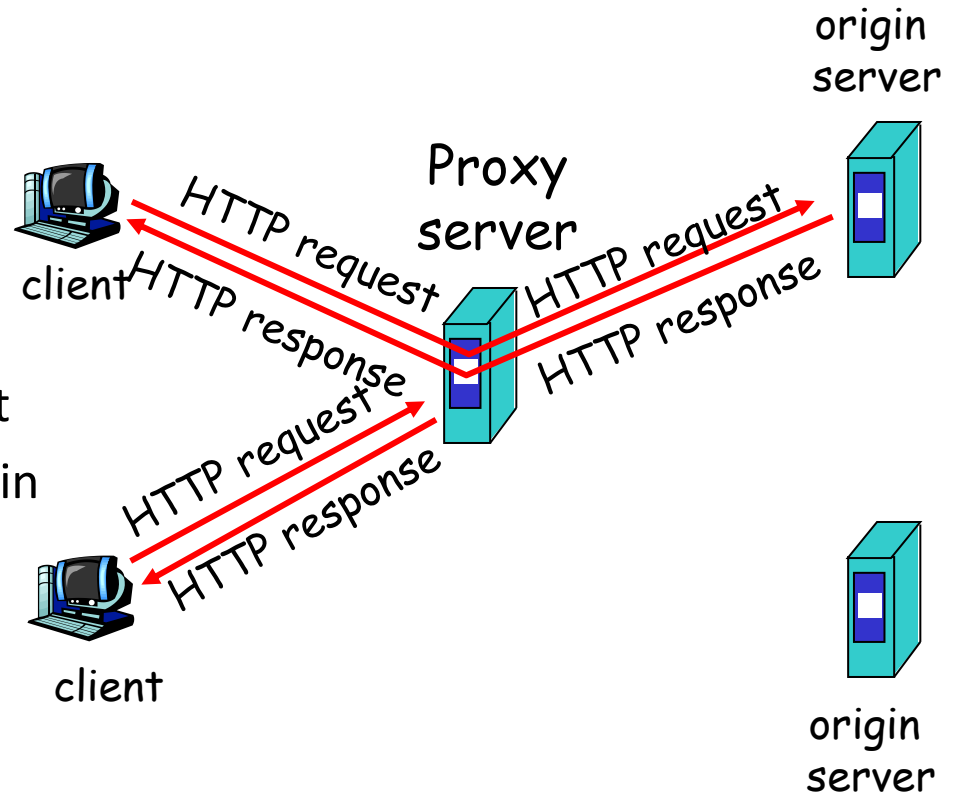
Connect to a web site (www.yahoo.com)

- Open and read the cookies.
- Modify them.
- Reconnect to the site.

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



Web caching:

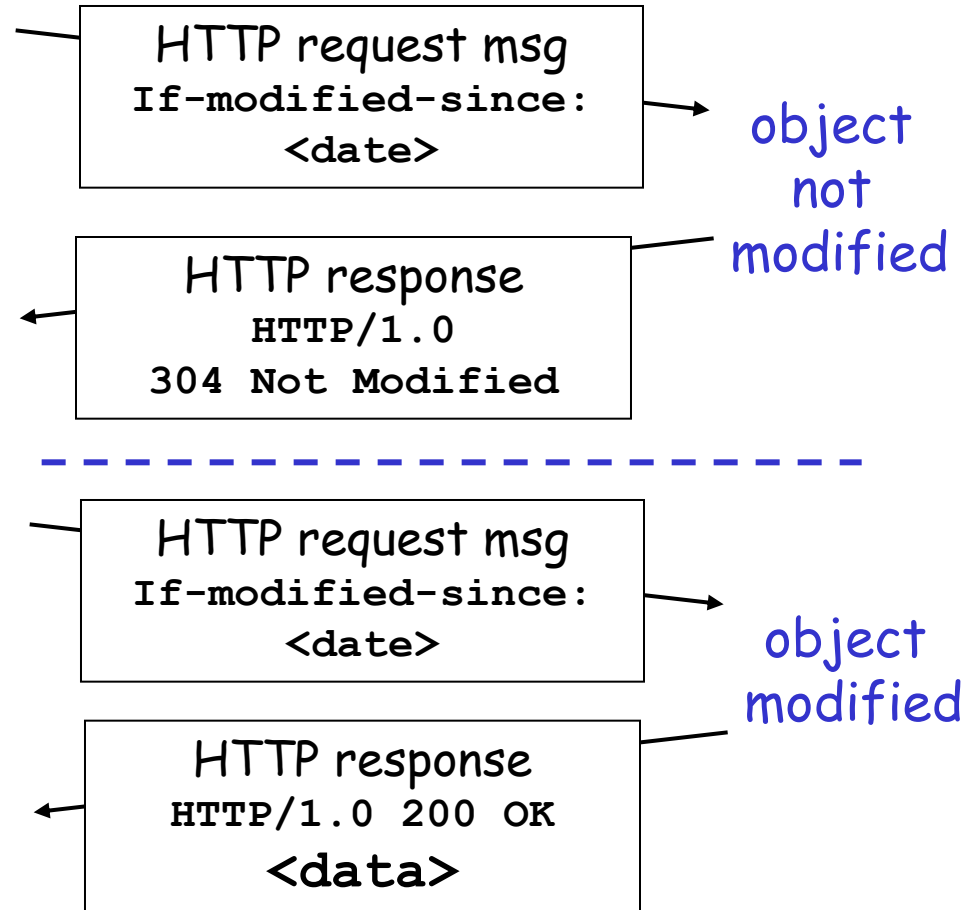
- Reduce response time for client request.
- Reduce traffic on an institution's access link.
- Internet dense with caches: enables “poor” content providers to effectively deliver content (but so does P2P file sharing)

Conditional get (used with caching)

- **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`

cache

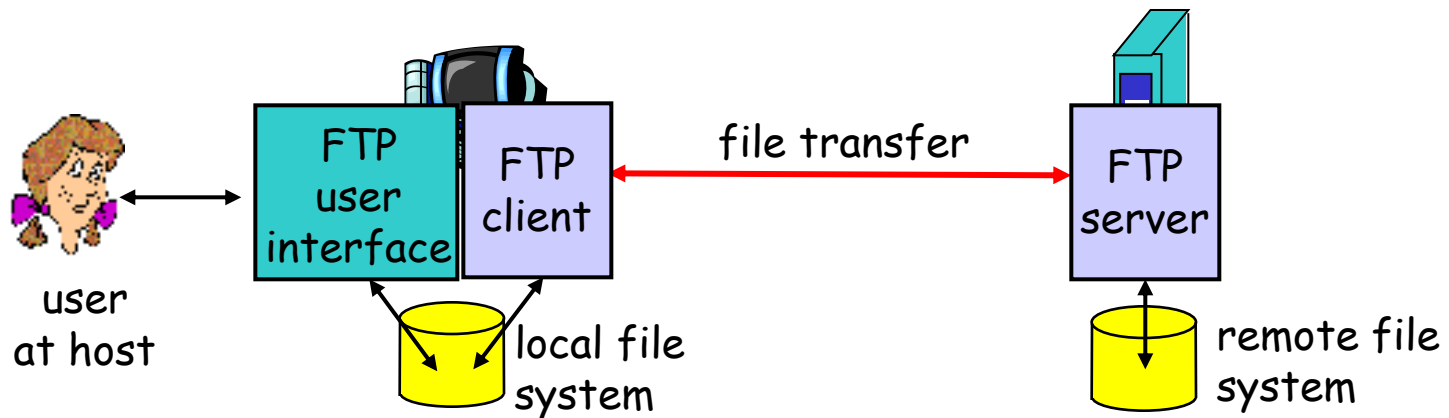
server



FTP

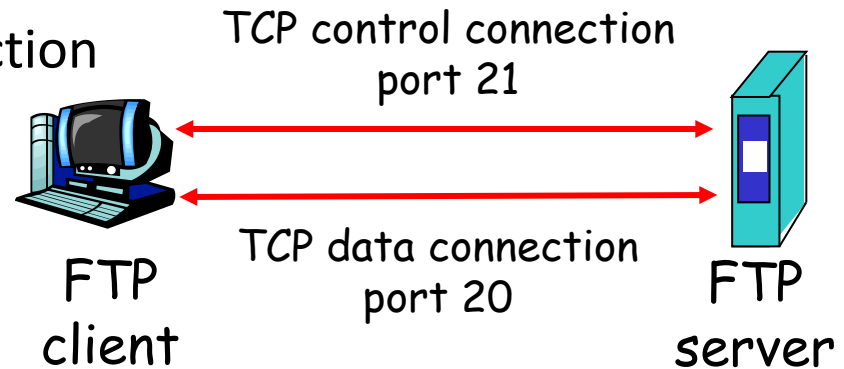
FTP: the file transfer protocol

- transfer file to/from remote host
- client/server model
 - *client*: side that initiates transfer (either to/from remote)
 - *server*: remote host
- ftp: RFC 959
- ftp server: **port 21** (control)



FTP: two connections, control & data

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection.
- When server receives file transfer command, server opens 2nd TCP connection (for file) to client
- After transferring one file, server closes data connection.
- Server opens another TCP data connection to transfer another file.
- Control connection: “out of band”
- FTP server maintains “state”: current directory, earlier authentication



FTP: commands - responses

Sample commands:

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

Sample return codes

status code and phrase (as in HTTP)

- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

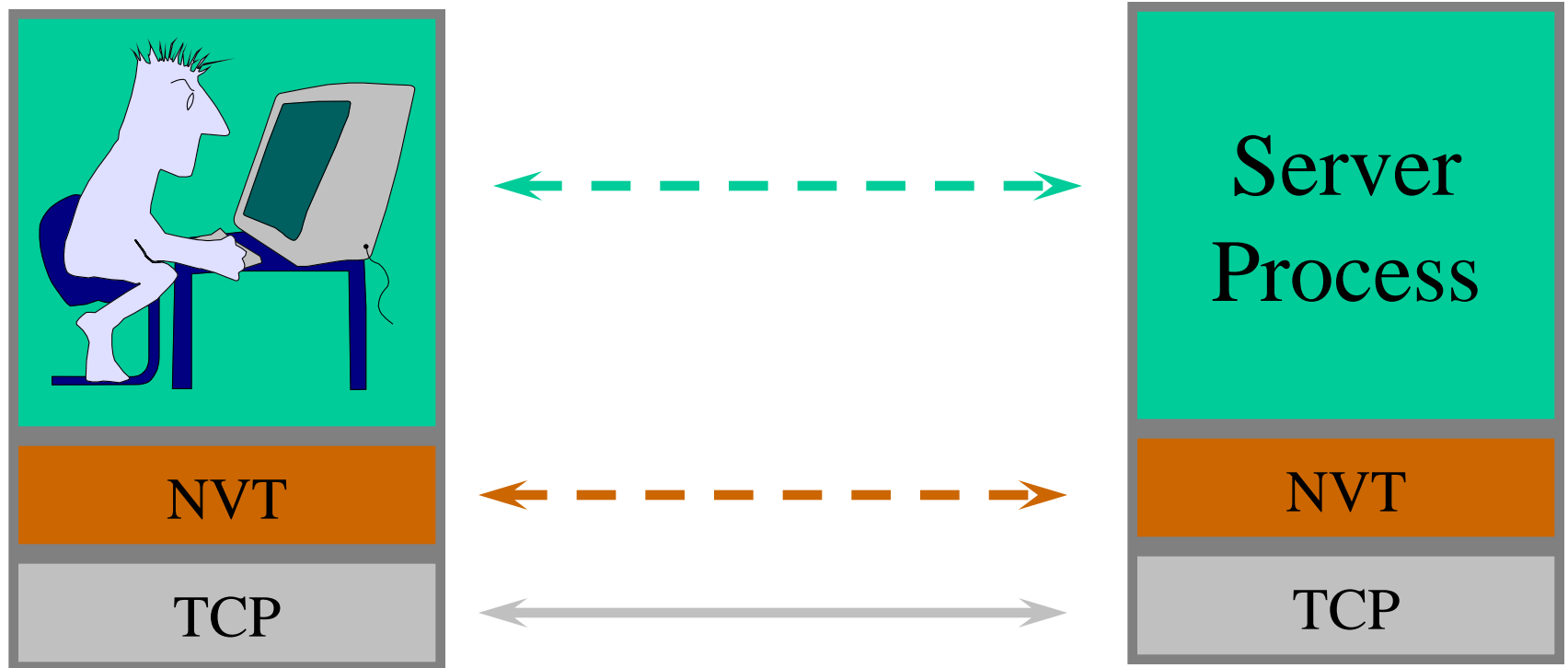
TELNET vs. telnet

- ❑ TELNET is a *protocol* that provides “a general, bi-directional, eight-bit byte oriented communications facility”.
- ❑ `telnet` is a *program* that supports the TELNET protocol over TCP.
- ❑ Many application protocols are built upon the TELNET protocol.

The TELNET Protocol

- ❑ Reference: RFC 854
- ❑ TCP connection
- ❑ data and control over the same connection.
- ❑ Network Virtual Terminal
 - ❖ intermediate representation of a generic terminal.
 - ❖ provides a standard language for communication of terminal control functions.

Network Virtual Terminal



Playing with TELNET

- ❑ You can use the `telnet` program to play with the TELNET protocol.
- ❑ `telnet` is a *generic* TCP client.
 - ❖ Sends whatever you type to the TCP socket.
 - ❖ Prints whatever comes back through the TCP socket
 - ❖ Useful for testing TCP servers (ASCII based protocols).
- ❑ Many Unix systems have these servers running (by default):

❖ <code>echo</code>	port 7	<code>discard</code>	port 9
❖ <code>daytime</code>	port 13	<code>chargen</code>	port 19