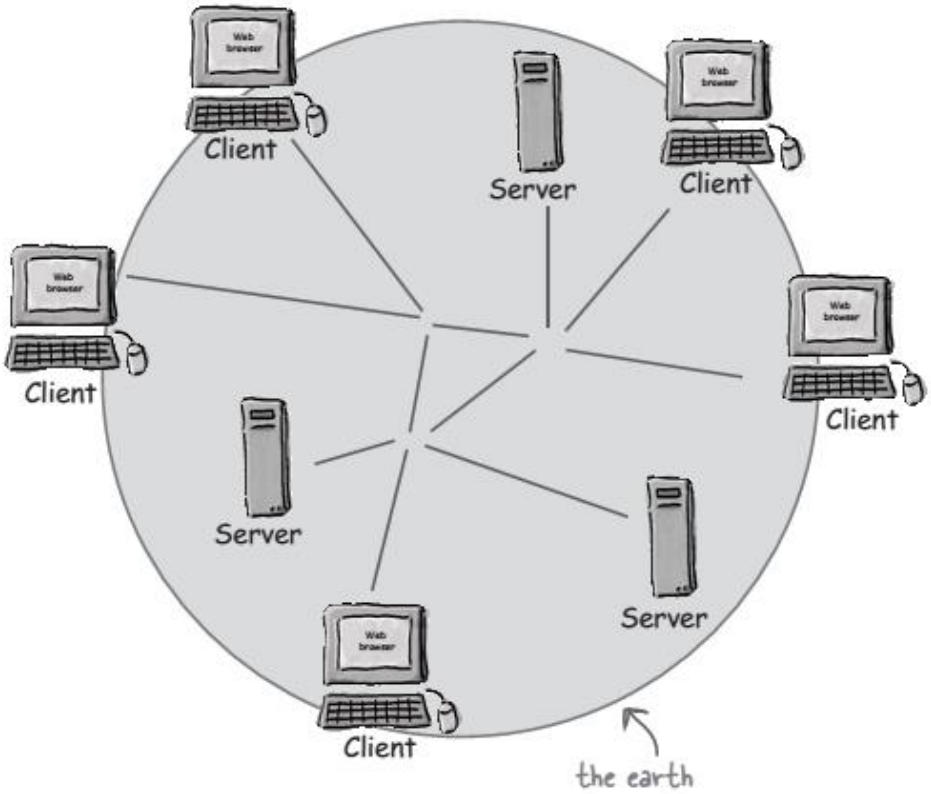
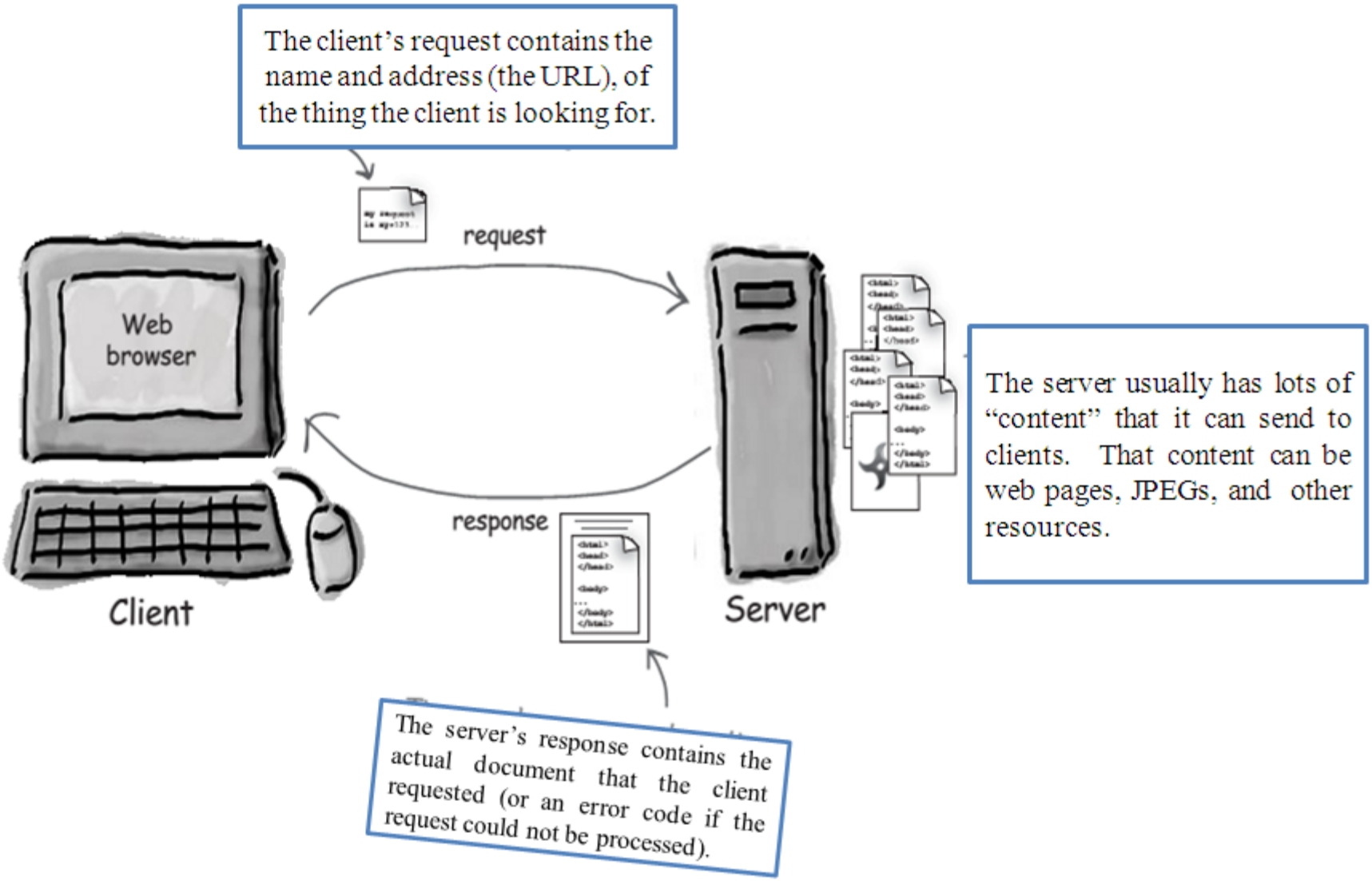
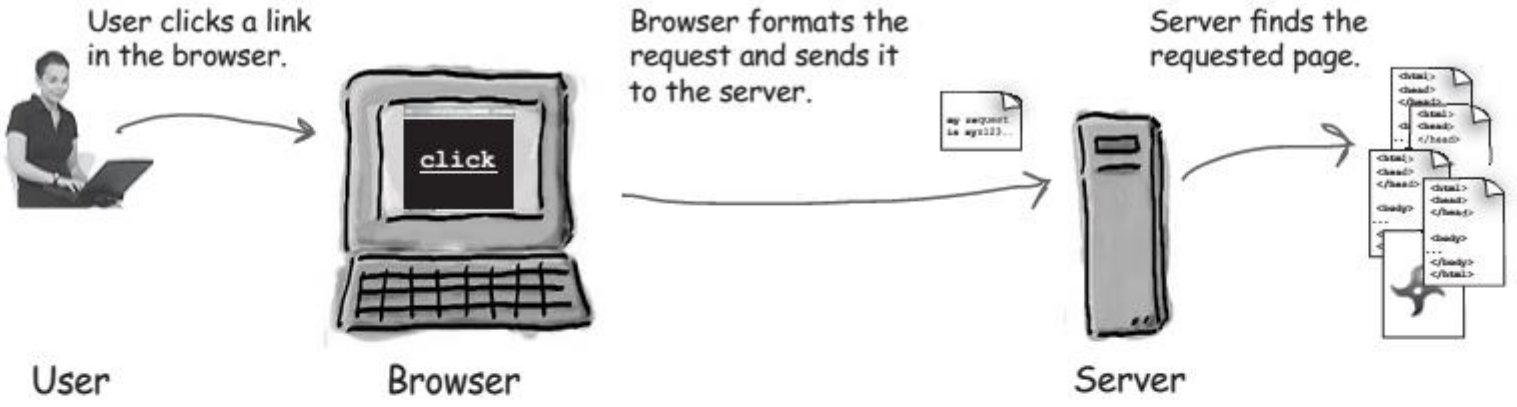
Web Engineering

HTTP Protocol

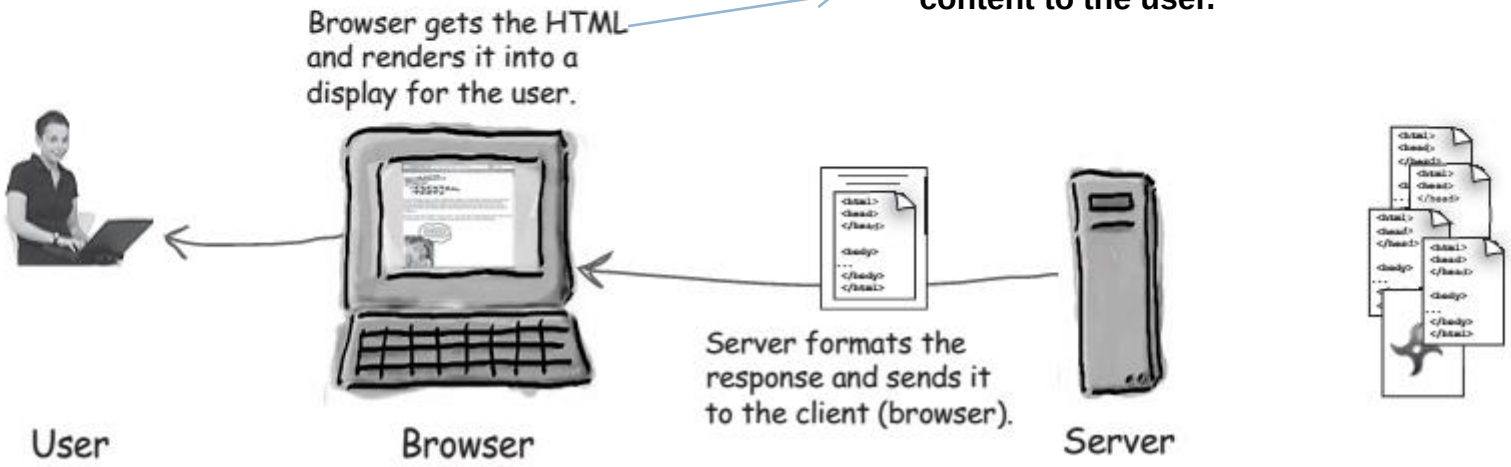
Internet and Web




The client's request contains the name and address (the URL), of the thing the client is looking for.
request
Web browser
Client
response
Server
The server usually has lots of "content" that it can send to clients. That content can be web pages, JPEGs, and other resources.
The server's response contains the actual document that the client requested (or an error code if the request could not be processed).



HTML tells the browser how to present the content to the user.



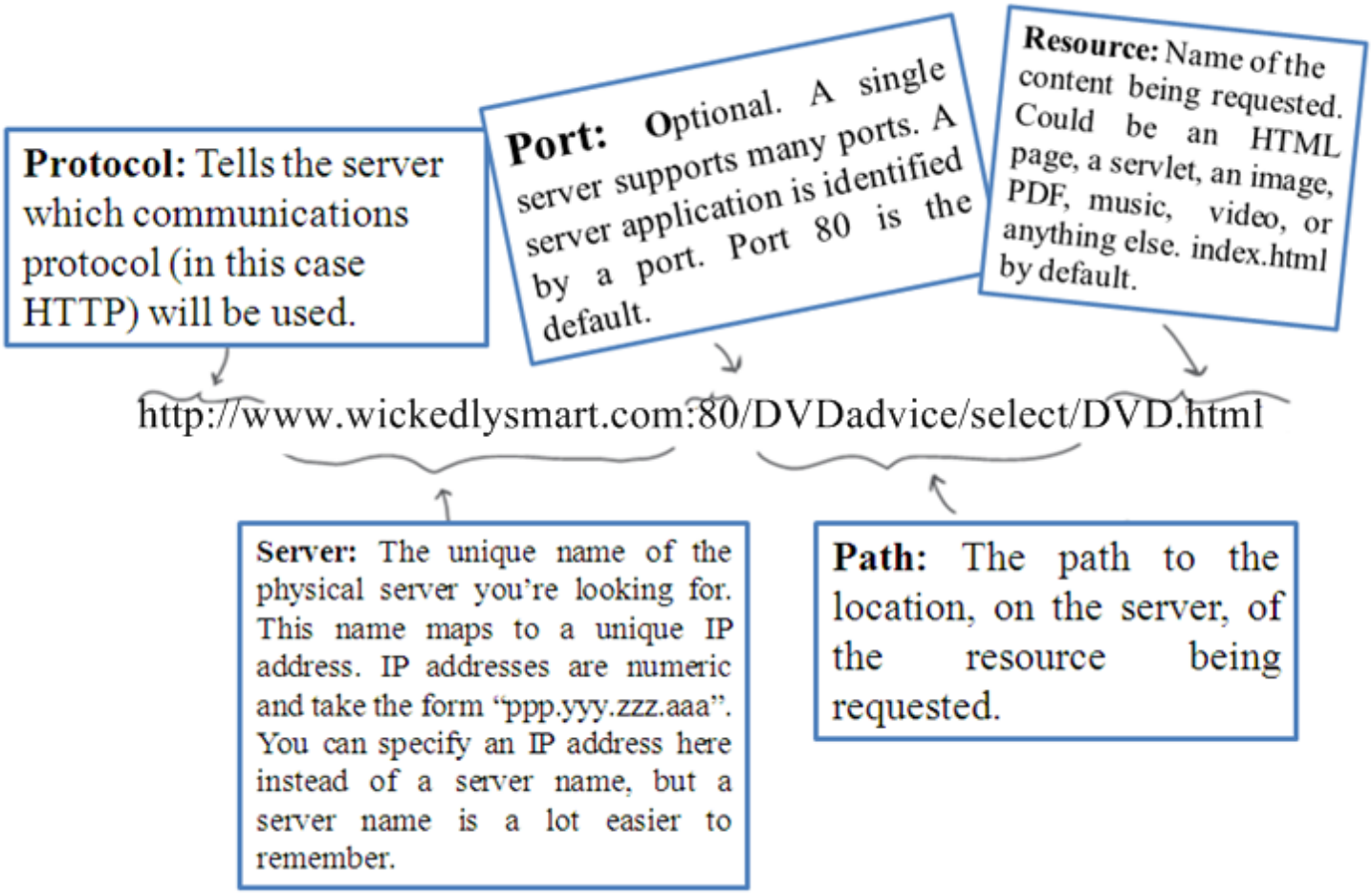
Web and HyperText Transfer Protocol (HTTP)

First some jargon

- 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:

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

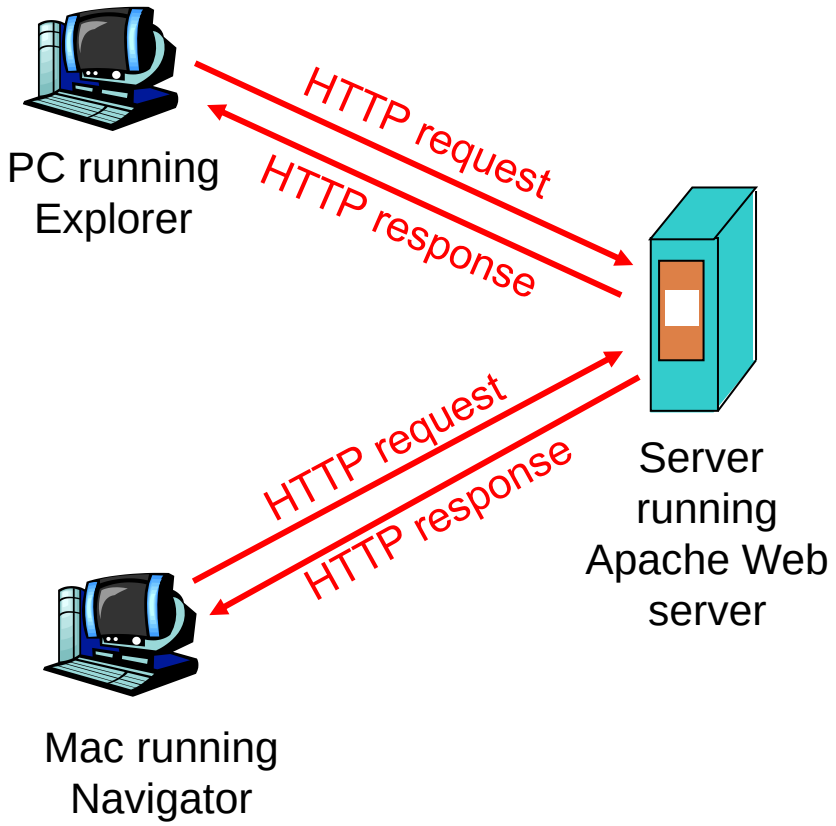
URL


Protocol: Tells the server which communications protocol (in this case HTTP) will be used.
Port: Optional. A single server supports many ports. A server application is identified by a port. Port 80 is the default.
Resource: Name of the content being requested. Could be an HTML page, a servlet, an image, PDF, music, video, or anything else. index.html by default.
http://www.wickedlysmart.com:80/DVDadvice/select/DVD.html
Server: The unique name of the physical server you're looking for. This name maps to a unique IP address. IP addresses are numeric and take the form "ppp.yyy.zzz.aaa". You can specify an IP address here instead of a server name, but a server name is a lot easier to remember.
Path: The path to the location, on the server, of the resource being requested.

HTTP overview

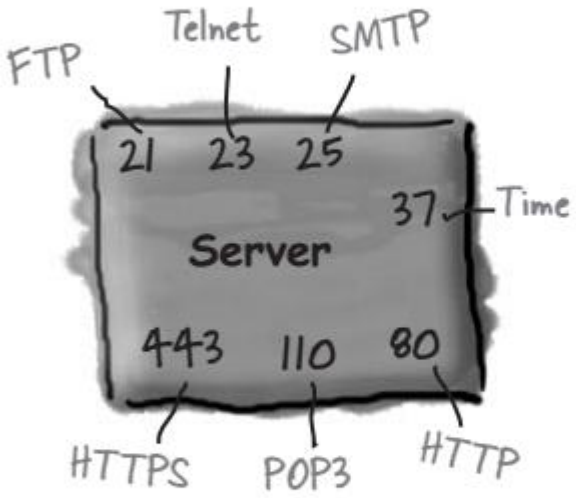
HTTP: hypertext transfer protocol

- 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



Ports

Well-known TCP port numbers for common server applications



Using one server app per port, a server can have up to 65536 different server apps running.

- The TCP port numbers from 0 to 1023 are reserved for well-known services.
- Don't use these ports for your own custom server programs!

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.
- 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

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) at www.someSchool.edu on port 80

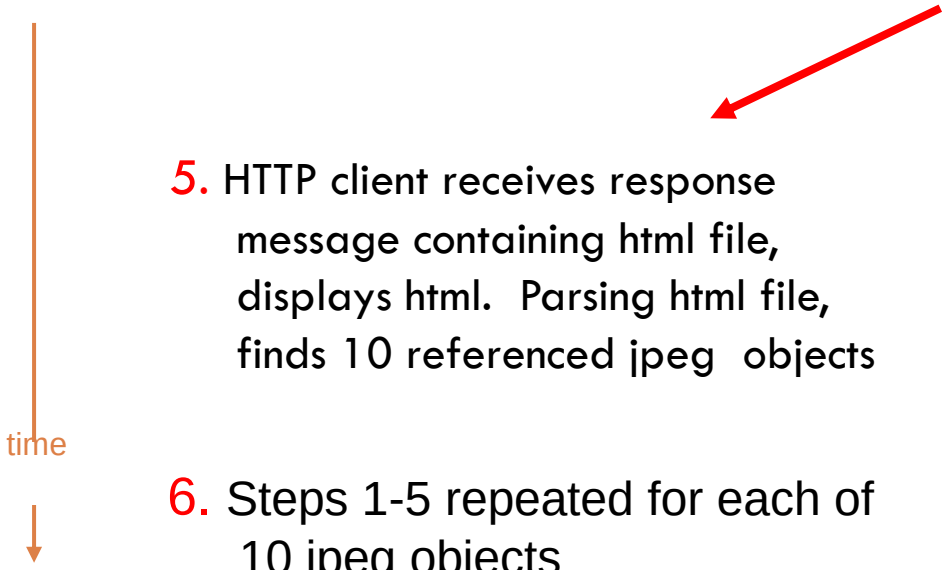
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

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.)



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

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

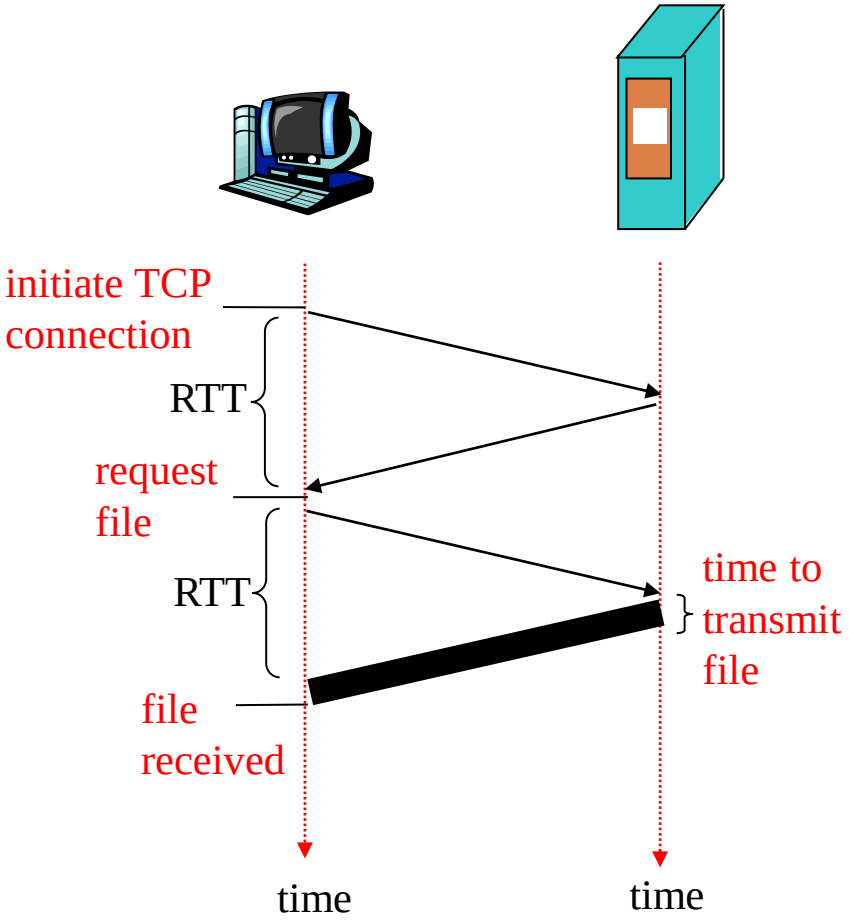
Response time modeling

Definition of RRT: 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+transmit time



Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- OS must work and allocate host resources for each TCP connection
- but 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 are sent over connection

Persistent without pipelining:

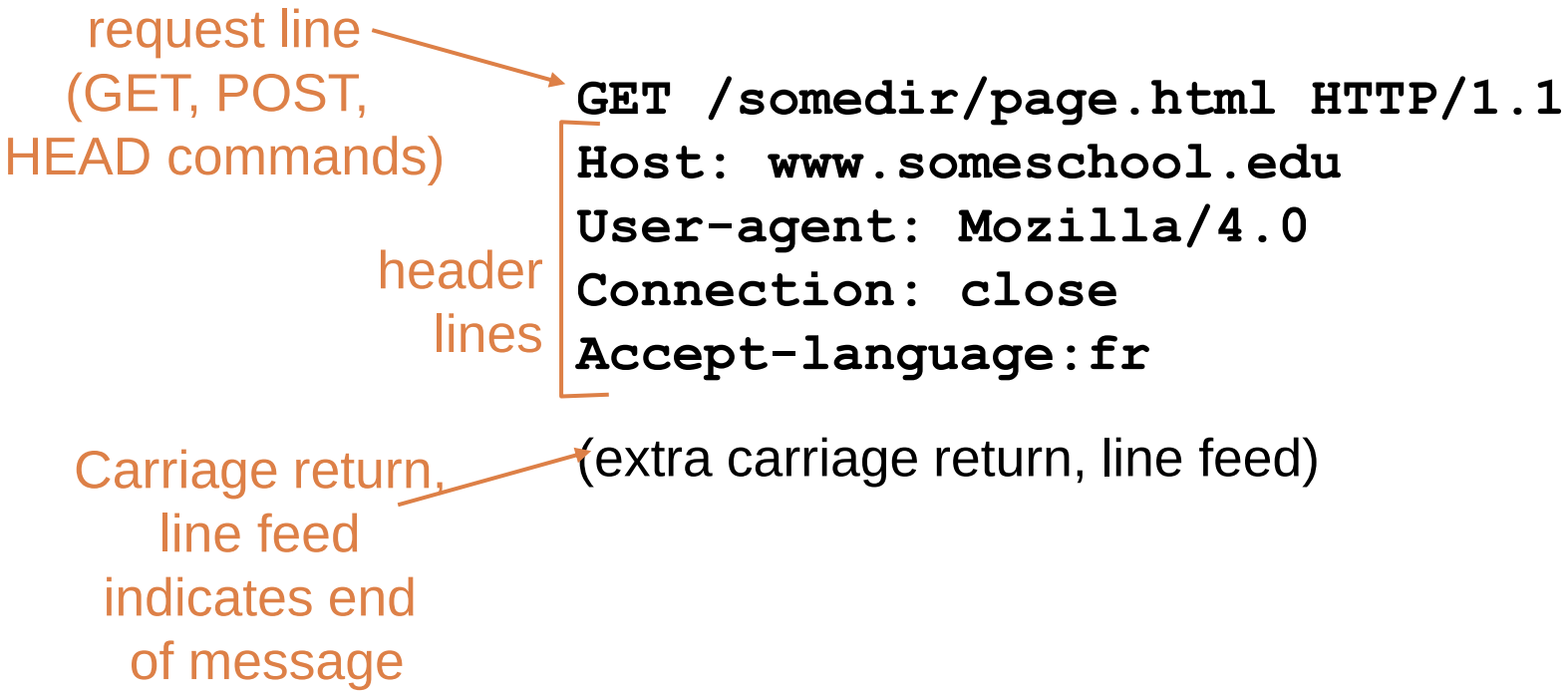
- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:

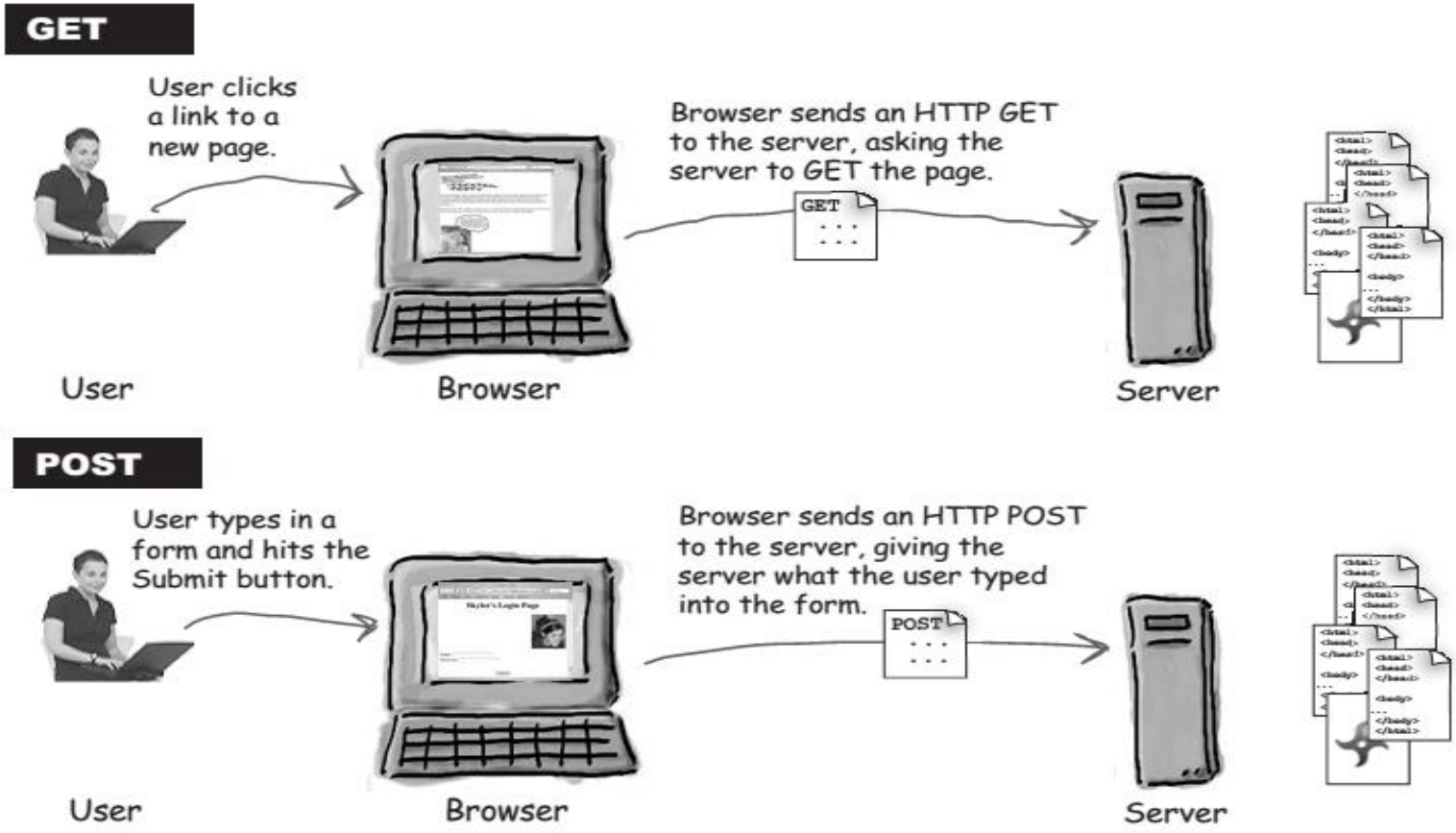
- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

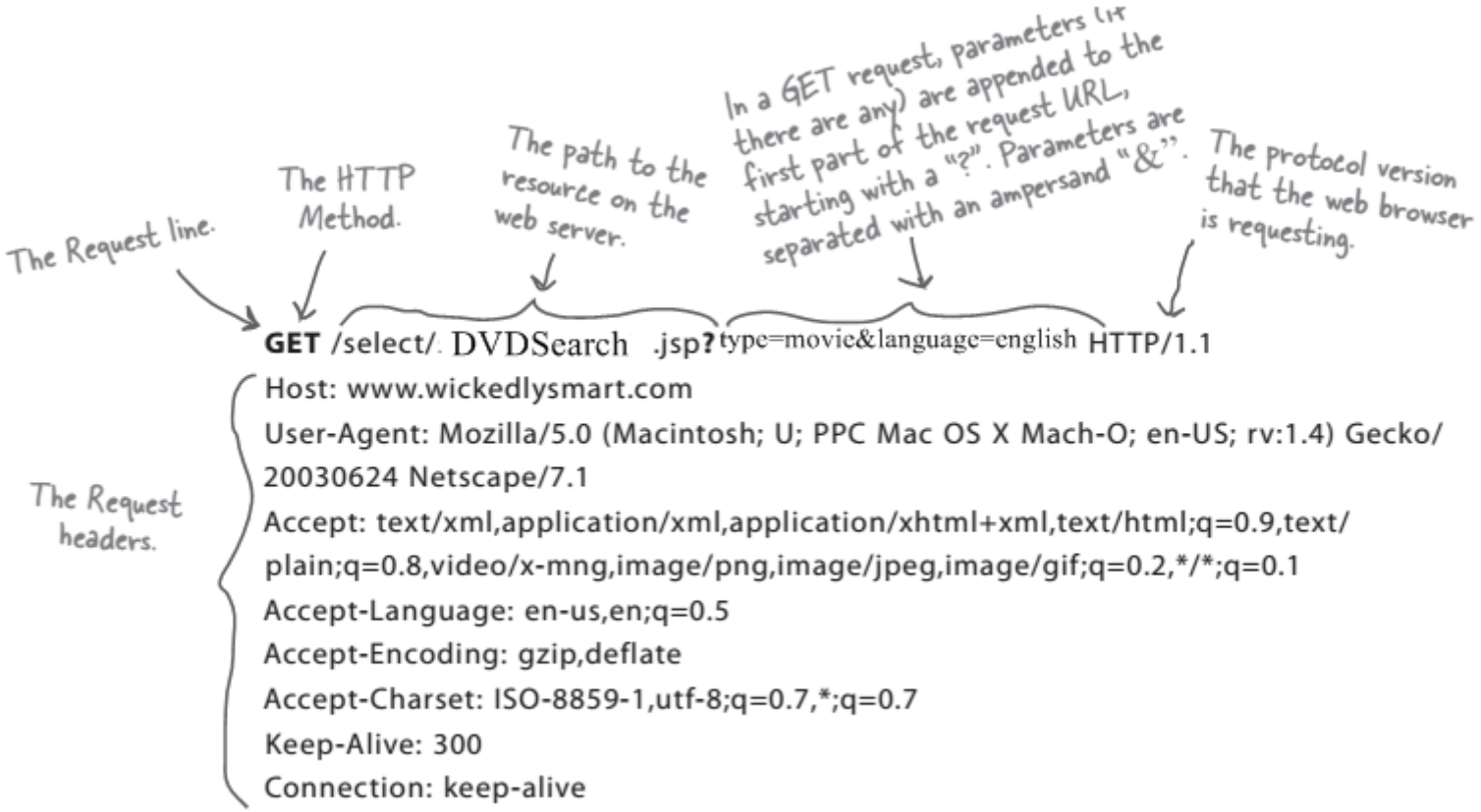
- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)



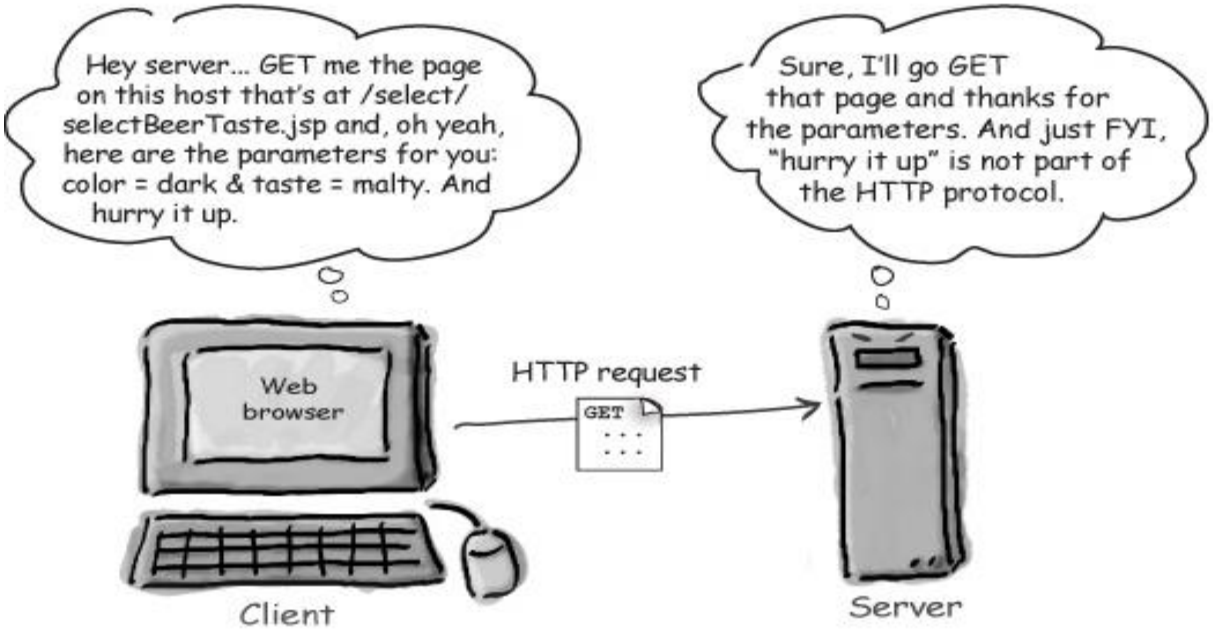
HTTP request message



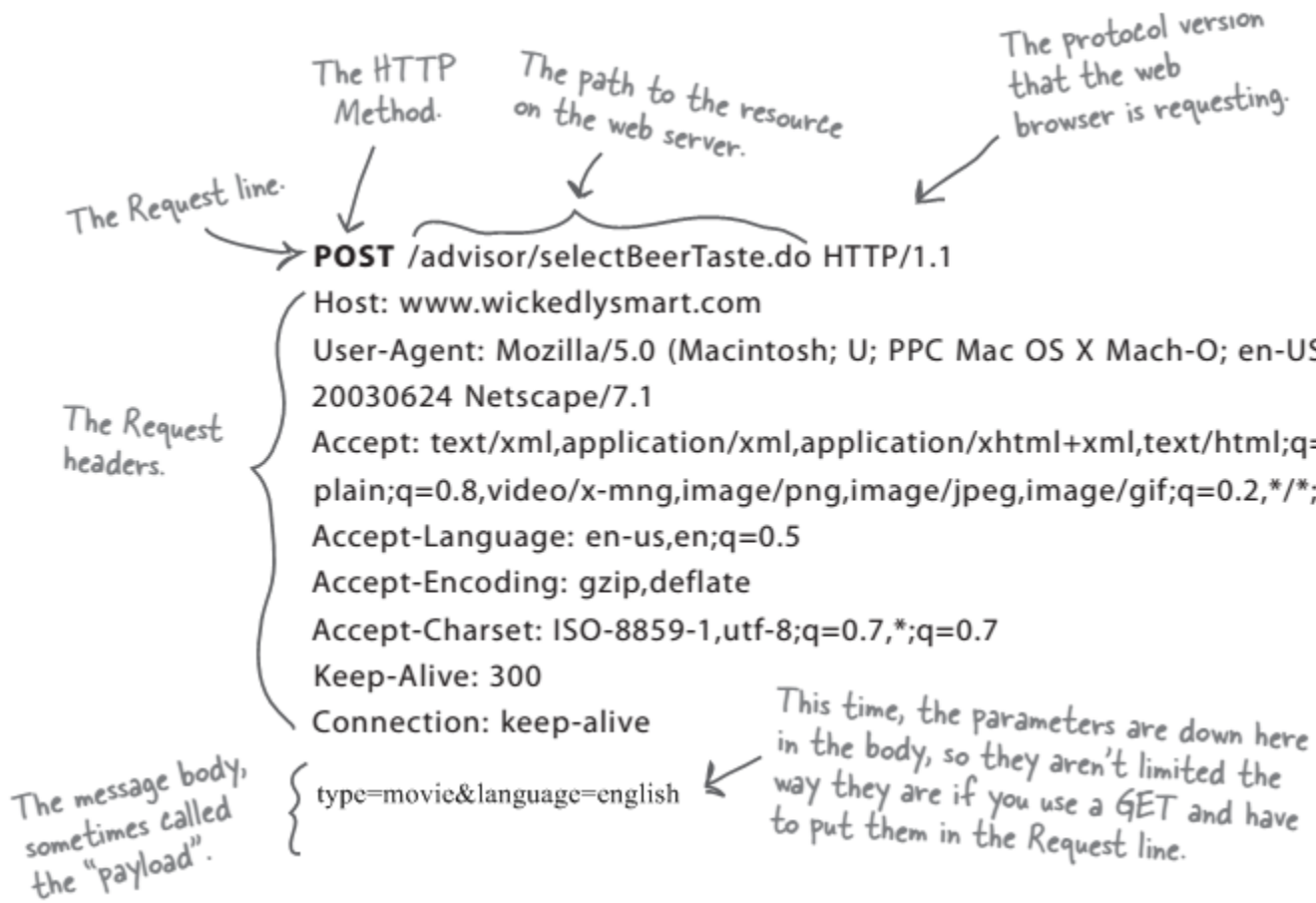
Anatomy of an HTTP GET request



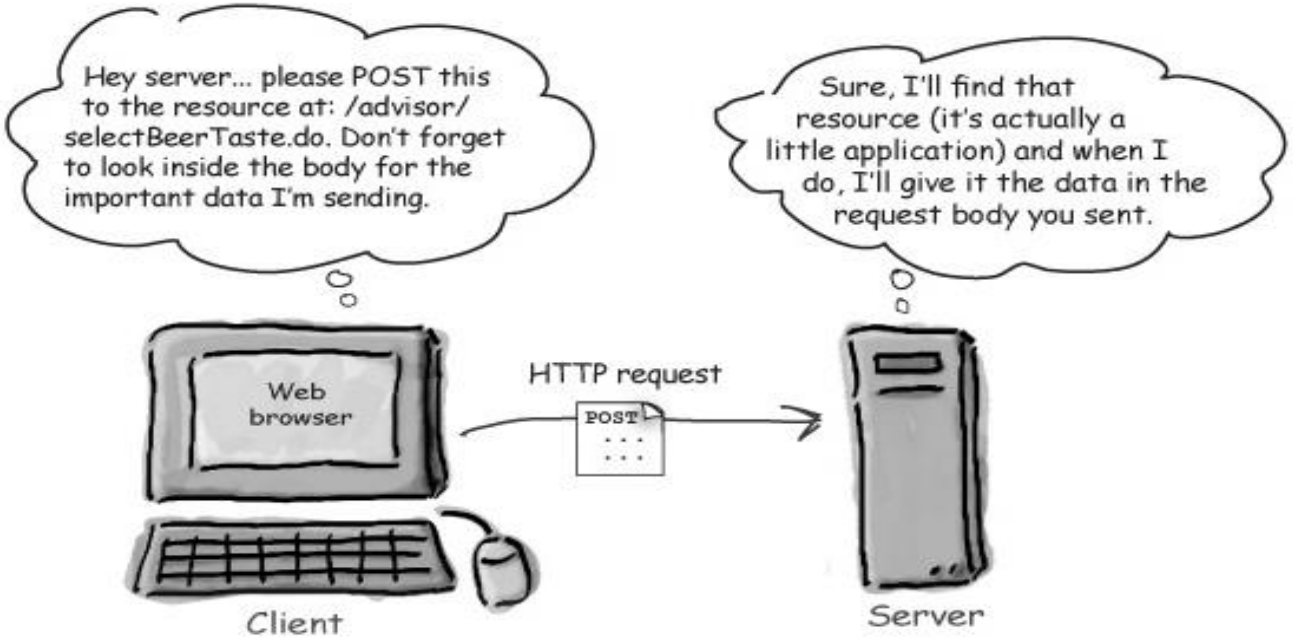
Anatomy of an HTTP GET request



Anatomy of an HTTP POST request



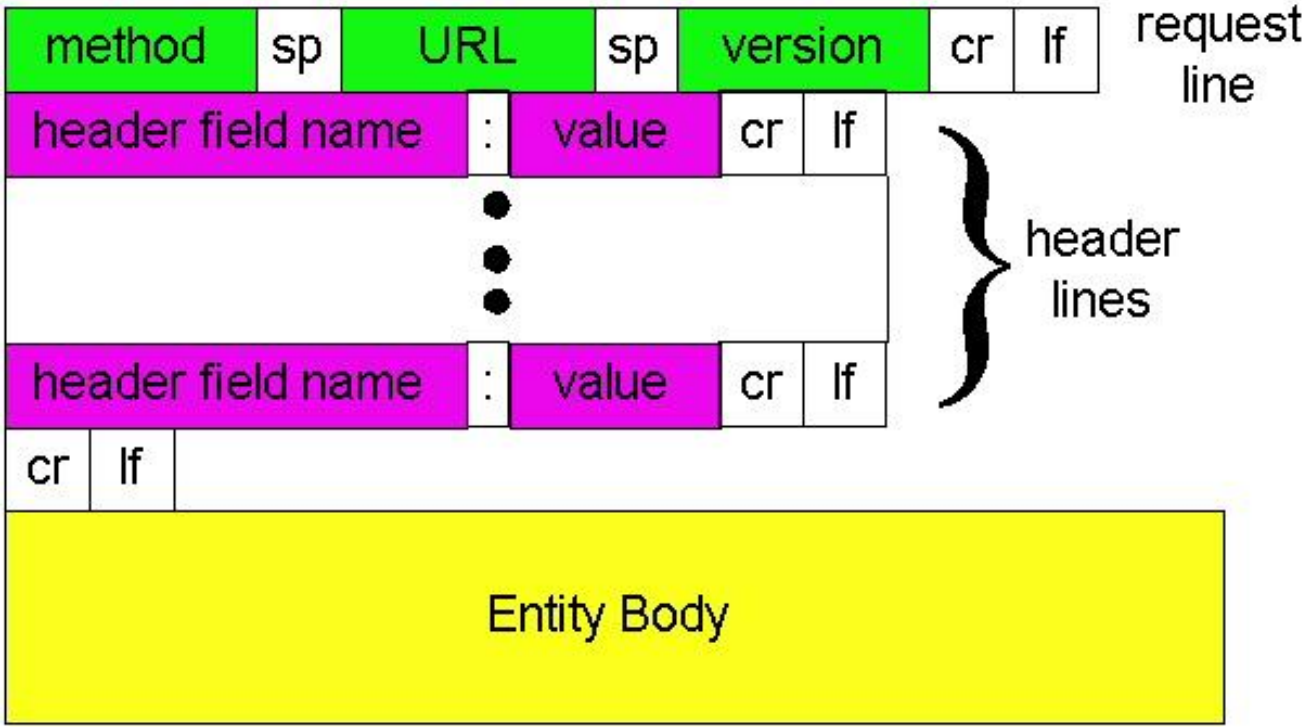
Anatomy of an HTTP POST request



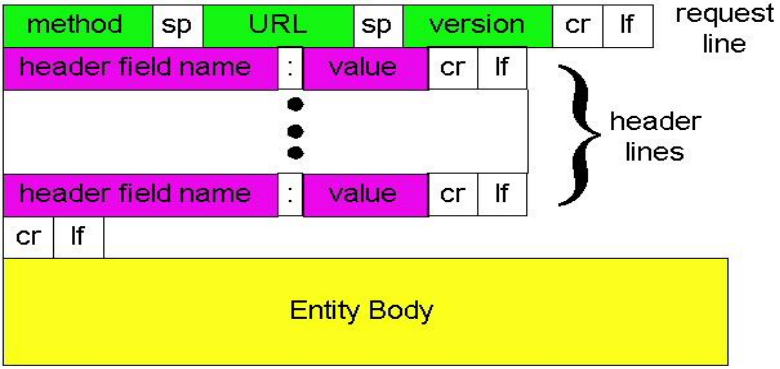
HTTP request message: general format

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

(extra carriage return, line feed)



HTTP request message: general format



Now let's look at the header lines in the example. The header line HOST: www.someschool.edu specifies the host on which the object resides. You night think that this header line is unnecessary, as there is already a TCP connection in place to the host. But, as we'll see in Section 2.2.6, the information provided by the host header line is required by Web proxy caches. By including theConnection:close header line, the browser is telling the server that it doesn't want to use persistent connections; it wants the server to close the connection after sending the requested object. Thus the browser that generated this request message implements HTTP/1.1 but it doesn't want to bother with persistent connections. The User-agent: header line specifies the user agent, that is, the browser type that is making the request to the server . Here the user agent is Mozilla/4.0, a Netscape browser. This header line is useful because the server can actually send different versions of the same object to different types of user agents. (Each of the versions is addressed by the same URL.) Finally, the Accept-language: header indicates that the user prefers to receive a French version of the object, if such an object exists on the server; otherwise, the server should send its default version.

The Entity Body is not used with the GET method, but is used with the POST method. The HTTP client uses the POST method when the user fills out a form

Method types

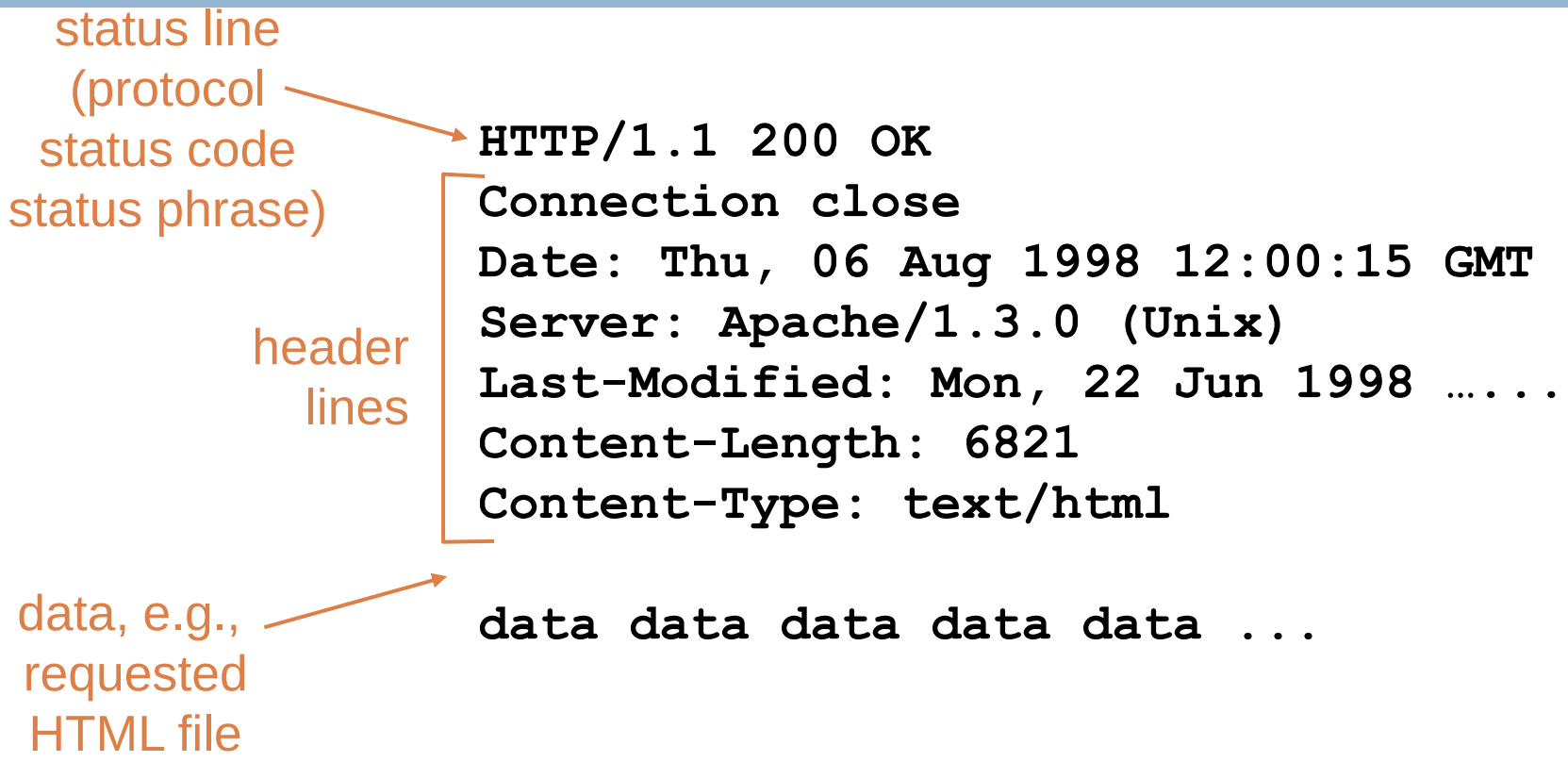
HTTP/1.0

- 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

User-Server Interaction: Authorization and Cookies

- HTTP server is stateless – simplifies server design
- Sometime server needs to identify user
- Two mechanism for identification:

  1. Authorization & 2. CooKies

Authorization :

1) Provide username and password to access documents on server
2) Status code 401: Authorization Required

User-server state: cookies

Many major Web sites use cookies

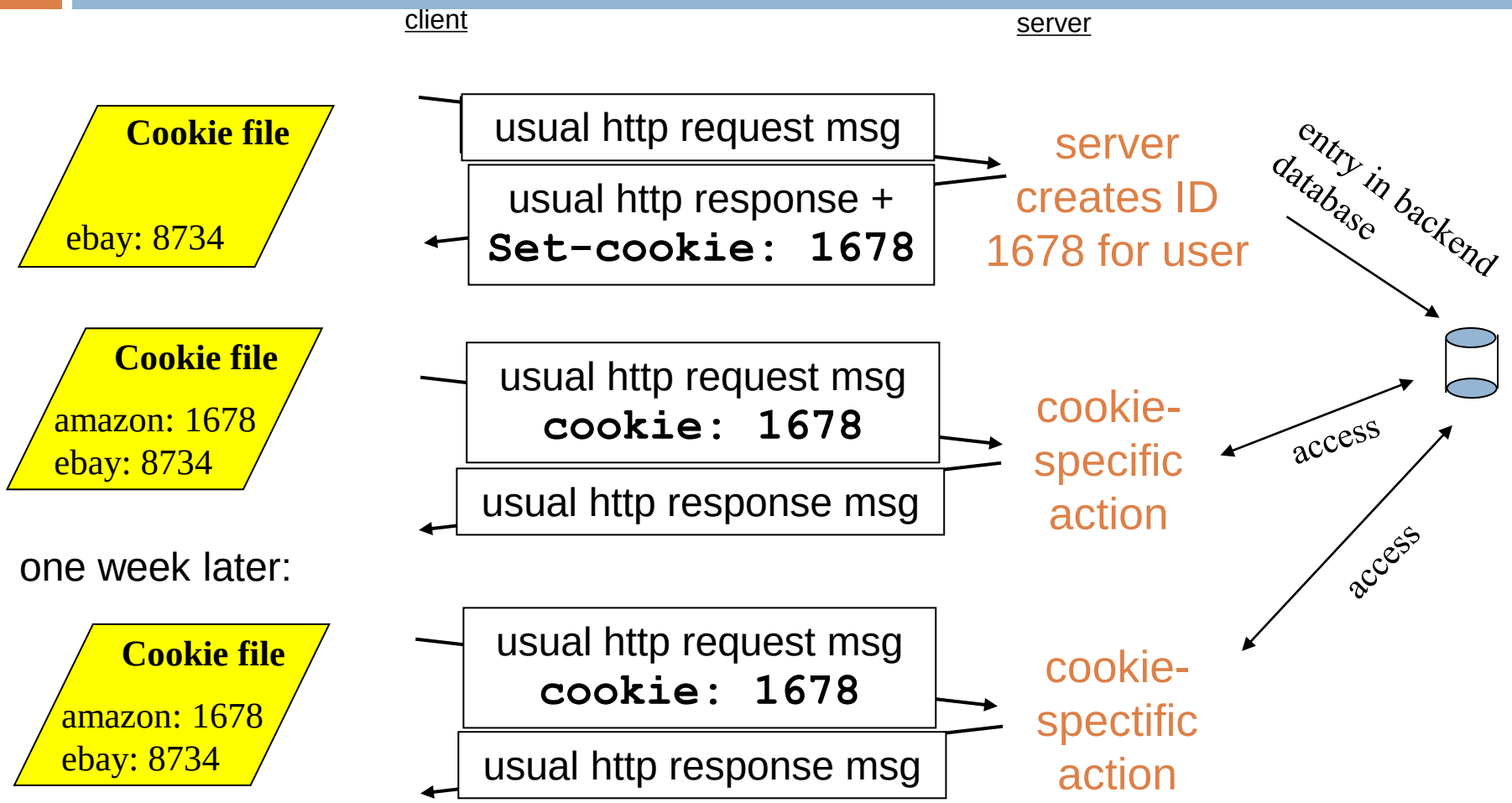
Four components:

1) cookie header line in the HTTP response message
2) cookie header line in HTTP request message
3) cookie file kept on user's host and 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

Cookies: keeping "state" (cont.)



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
- search engines use redirection & cookies to learn yet more
- advertising companies obtain info across sites

Thank you