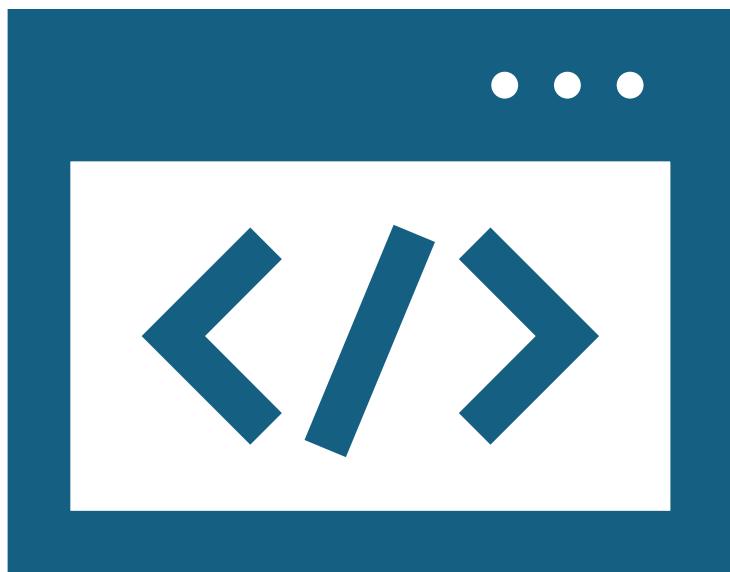


Part II: The Application Layer

Example: The World Wide Web

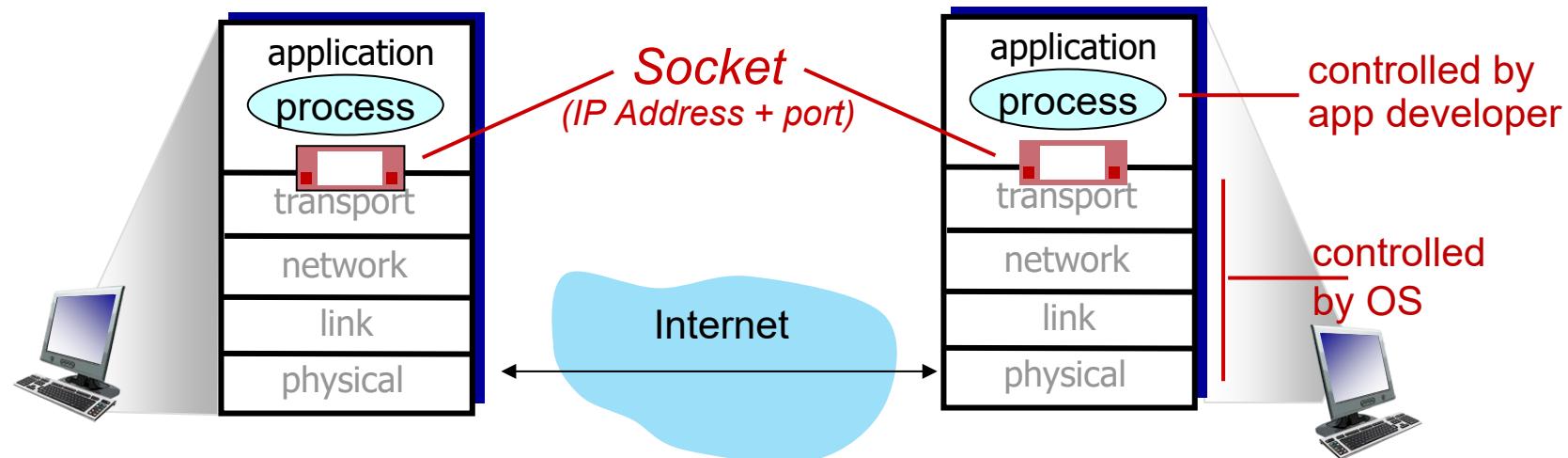


Summary

1. Sockets
2. Web and HTTP Overview
3. The Client Side
4. The Server Side
5. HTTP

Sockets

- process sends/receives messages to/from its **socket, identified by IP address and port number**
- socket analogous to door
 - sending process shoves message out of the door
 - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
 - two sockets involved: one on each side



Web and HTTP

First, a quick review...

- The web consists of *objects*, each of which can be stored on different Web servers
- object can be HTML file, JPEG image, Java applet, audio file,...
- web page consists of *base HTML-file* which includes *several referenced objects, each* addressable by a *URL*, e.g.,

www.someschool.edu/someDept/pic.gif

The URL 'www.someschool.edu/someDept/pic.gif' is shown. A horizontal brace is positioned below the first part of the URL, 'www.someschool.edu', with the label 'host name' centered below it. Another horizontal brace is positioned below the remaining part of the URL, '/someDept/pic.gif', with the label 'path name' centered below it.

URLs – Uniform Resource Locators

Some common URLs.

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending e-mail	mailto:JohnUser@acm.org
telnet	Remote login	telnet://www.w3.org:80

Web Overview

- (a) A Web page
- (b) The page reached by clicking on Department of Animal Psychology.

WELCOME TO THE UNIVERSITY OF EAST PODUNK'S WWW HOME PAGE

- Campus Information
 - [Admissions information](#)
 - [Campus map](#)
 - [Directions to campus](#)
 - [The UEP student body](#)
- Academic Departments
 - [Department of Animal Psychology](#)
 - [Department of Alternative Studies](#)
 - [Department of Microbiotic Cooking](#)
 - [Department of Nontraditional Studies](#)
 - [Department of Traditional Studies](#)

Webmaster@eastpodunk.edu

(a)

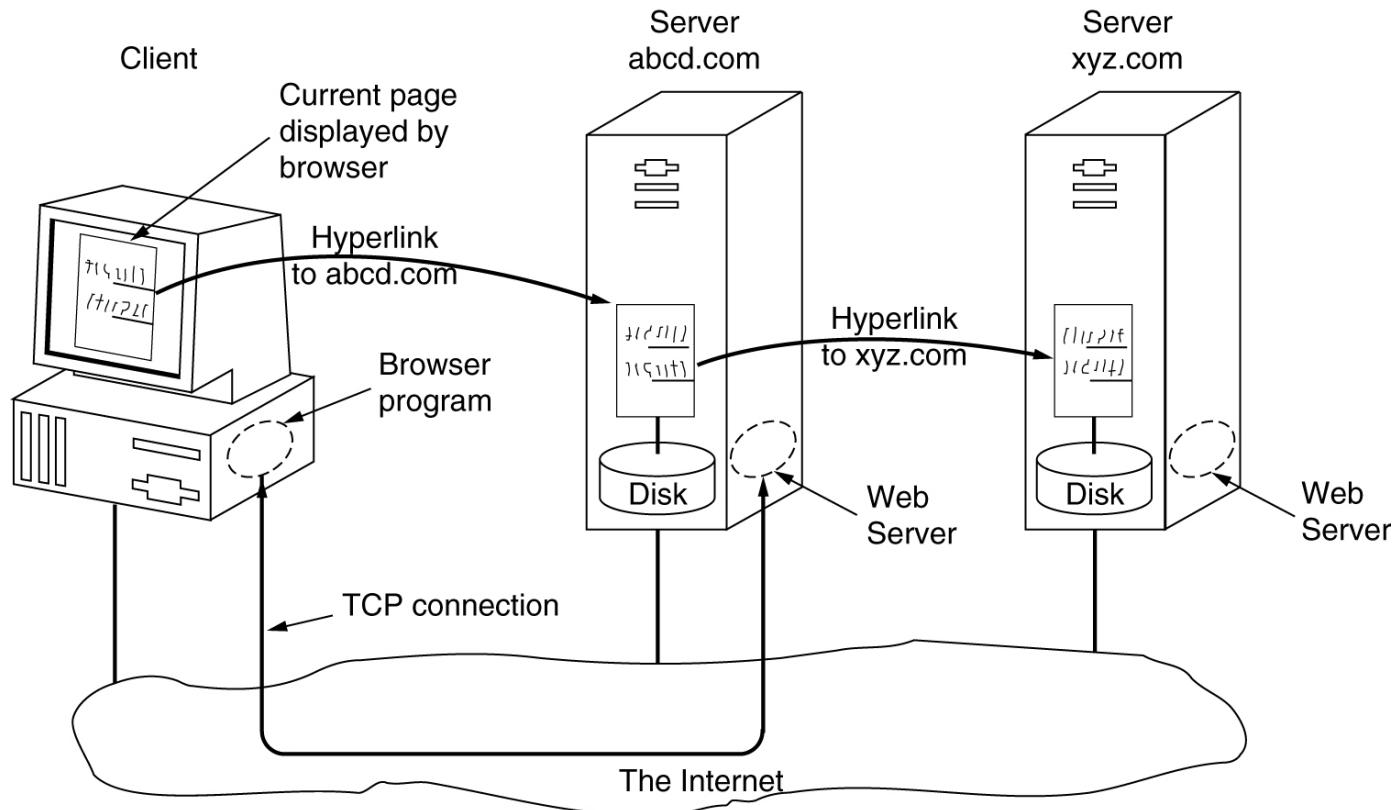
THE DEPARTMENT OF ANIMAL PSYCHOLOGY

- [Information for prospective majors](#)
- Personnel
 - [Faculty members](#)
 - [Graduate students](#)
 - [Nonacademic staff](#)
- [Research Projects](#)
- [Positions available](#)
- Our most popular courses
 - [Dealing with herbivores](#)
 - [Horse management](#)
 - [Negotiating with your pet](#)
 - [User-friendly doghouse construction](#)
- [Full list of courses](#)

Webmaster@animalpsyc.eastpodunk.edu

(b)

Web Overview: Client-Server Architecture



The parts of the Web model

HTTP overview

HTTP: hypertext transfer protocol

- Web's application-layer protocol
- client/server model:
 - *client*: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
 - *server*: Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

HTTP 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

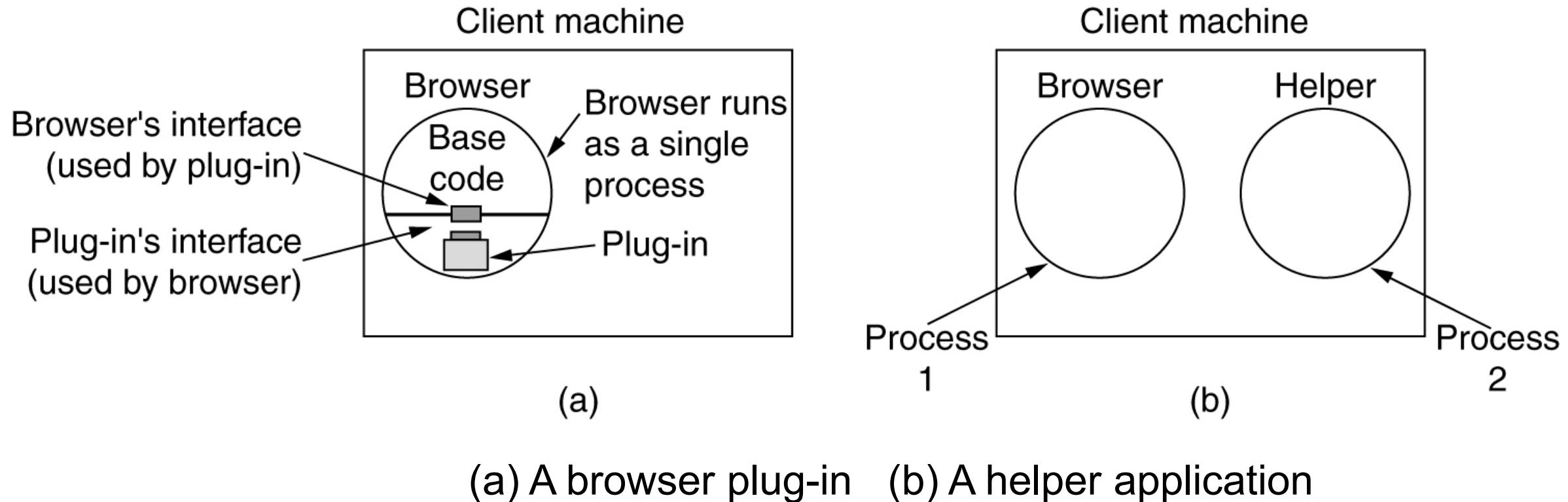
HTTP is “stateless”

- server maintains *no* information about past client requests

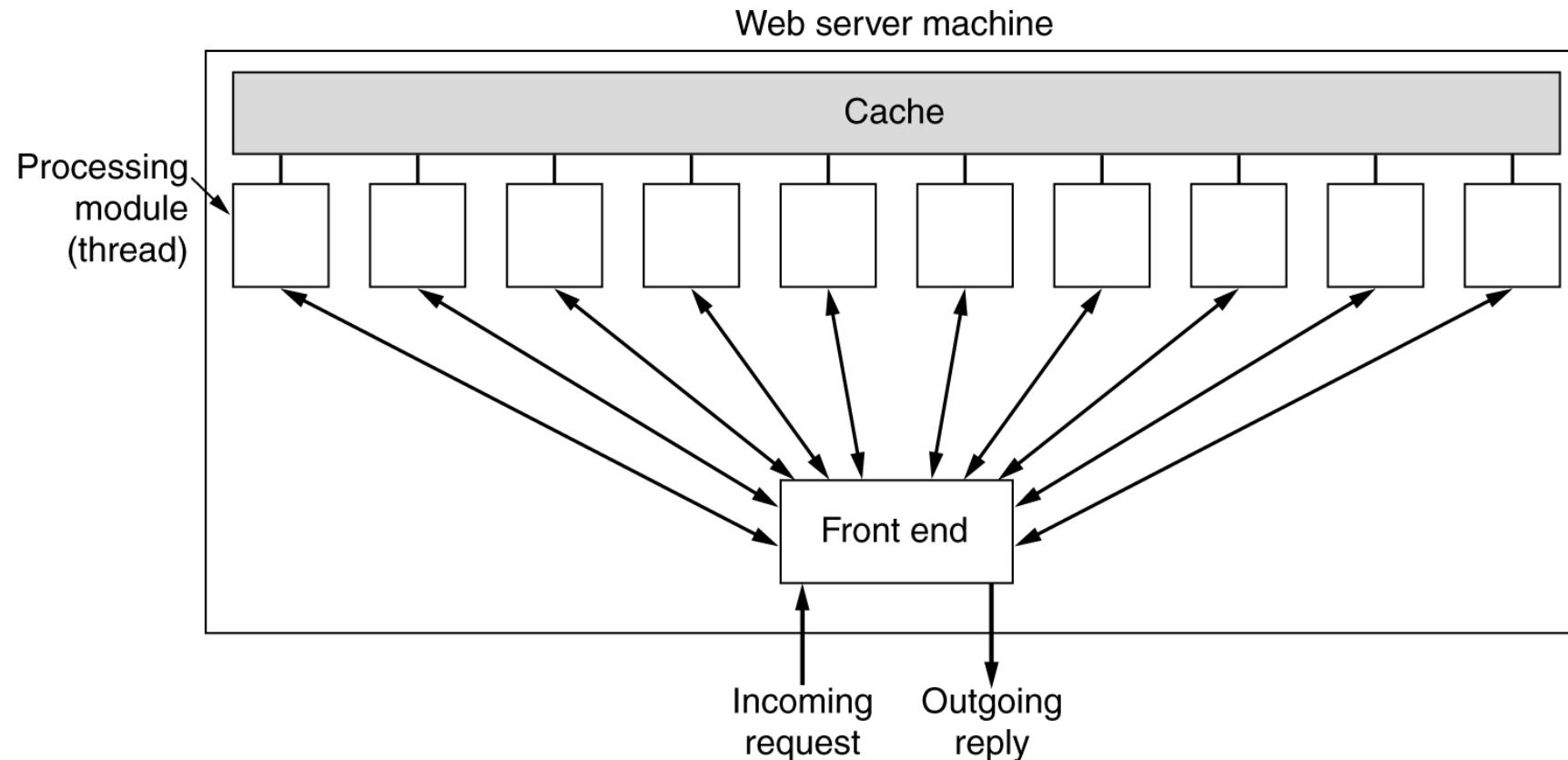
aside
protocols that maintain
“state” are complex!

- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled

The Client Side

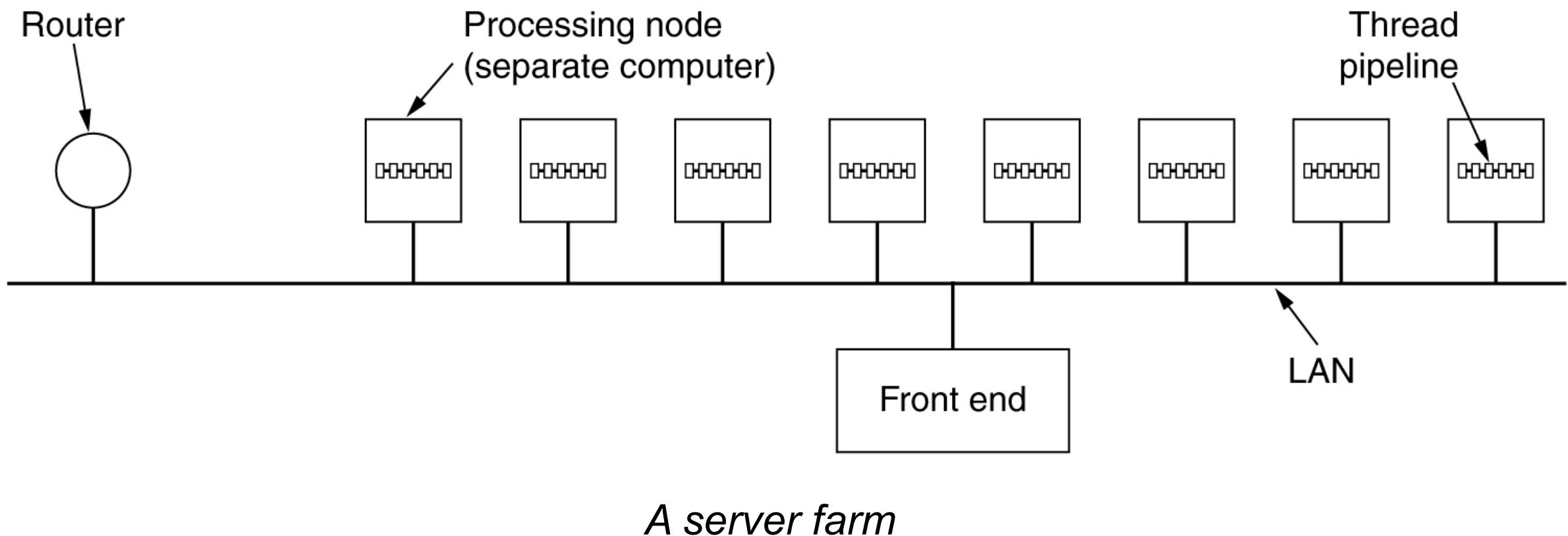


The Server Side (1)



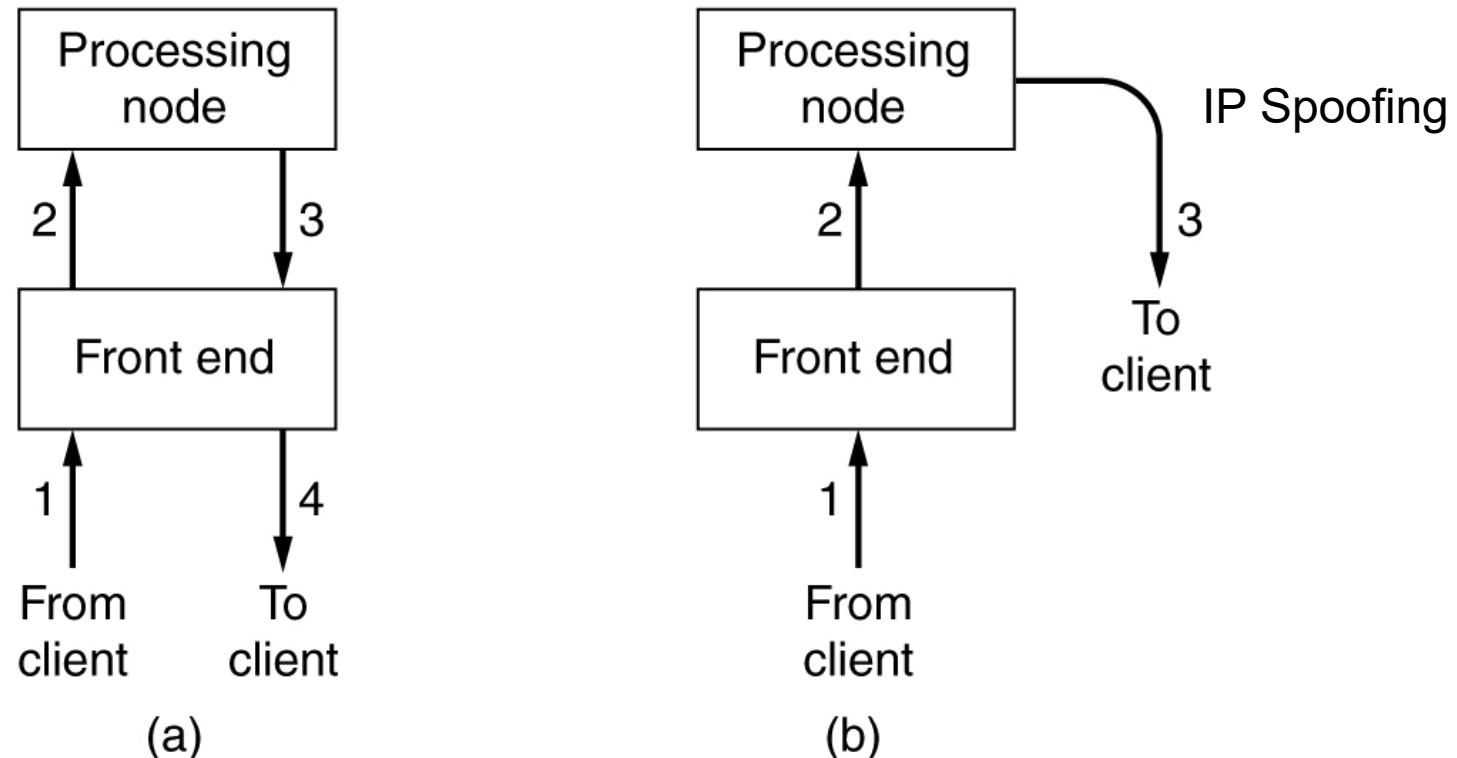
A multithreaded Web server with a front end and processing modules.

The Server Side (2)



A server farm

The Server Side (3)



- (a) Normal request-reply message sequence.
(b) Sequence when TCP handoff is used.

HTTP connections: two types

Non-persistent HTTP

1. TCP connection opened
2. at most one object sent over TCP connection
3. TCP connection closed

downloading multiple objects required multiple connections

Persistent HTTP

- TCP connection opened to a server
- multiple objects can be sent over *single* TCP connection between client, and that server
- TCP connection closed

Non-persistent HTTP: example

User enters URL: `www.someSchool.edu/someDepartment/home.index`
(containing 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

time
↓

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

Non-persistent HTTP: example (cont.)

User enters URL: `www.someSchool.edu/someDepartment/home.index`
(containing text, references to 10 jpeg images)



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



4. HTTP server closes TCP connection.

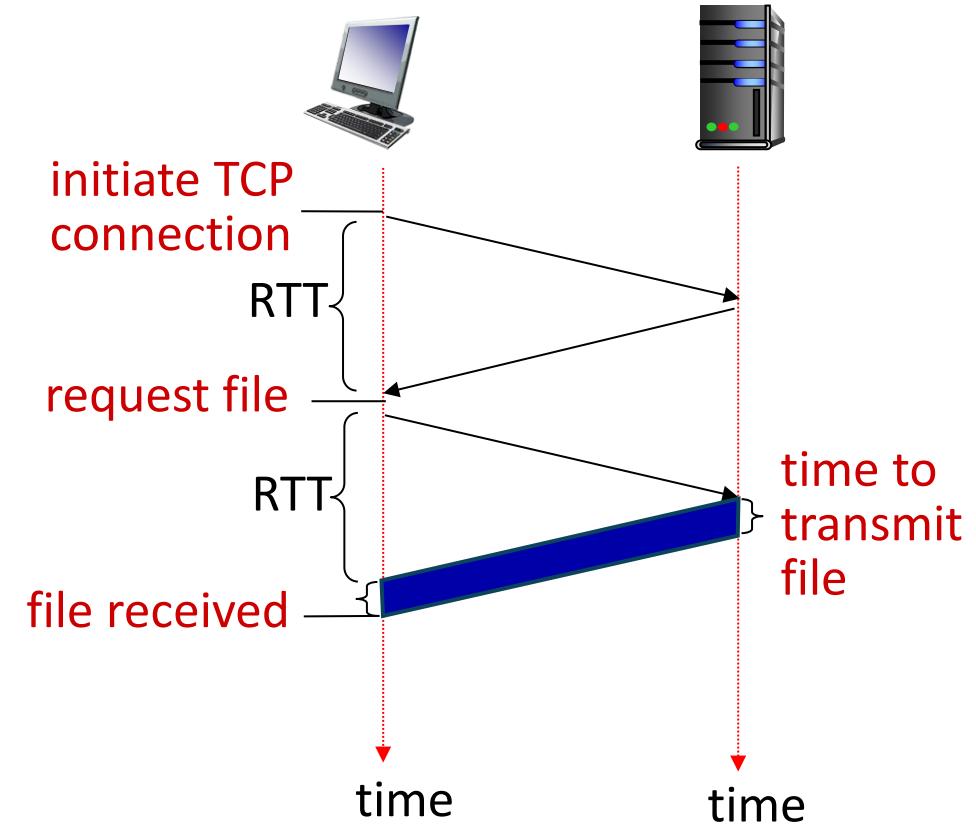
time

Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time (per object):

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



$$\text{Non-persistent HTTP response time} = 2\text{RTT} + \text{file transmission time}$$

Persistent HTTP (HTTP 1.1)

Non-persistent HTTP issues:

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

* Check out the online interactive animation for more
https://media.pearsoncmg.com/ph/esm/ecs_kurose_comnetwork_8/cw/#interactiveanimations

Persistent HTTP (HTTP1.1):

- 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 (cutting response time in half)

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

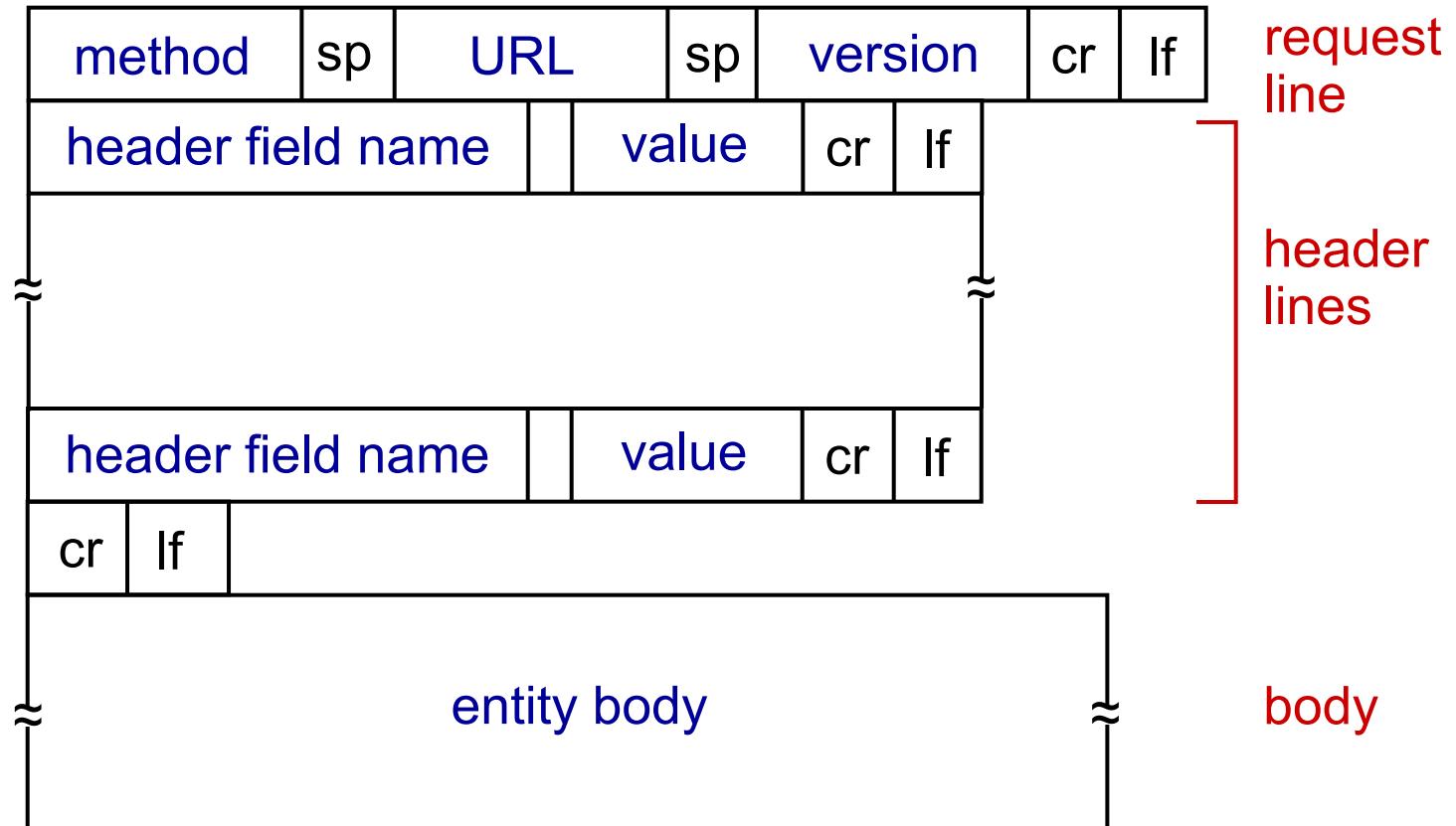
```
Host: www-net.cs.umass.edu\r\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X  
10.15; rv:80.0) Gecko/20100101 Firefox/80.0 \r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nConnection: keep-alive\r\n\r\n
```

carriage return character
line-feed character

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

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP request message: general format



Other HTTP request messages

POST method:

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a '?'):

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

HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested with an HTTP HEAD method.

PUT method:

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of PUT HTTP request message

HTTP response message

status line (protocol → **HTTP/1.1 200 OK**
status code status phrase)

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 message

301 Moved Permanently

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

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Maintaining user/server state: cookies

Web sites and client browser use *cookies* to maintain some state between transactions

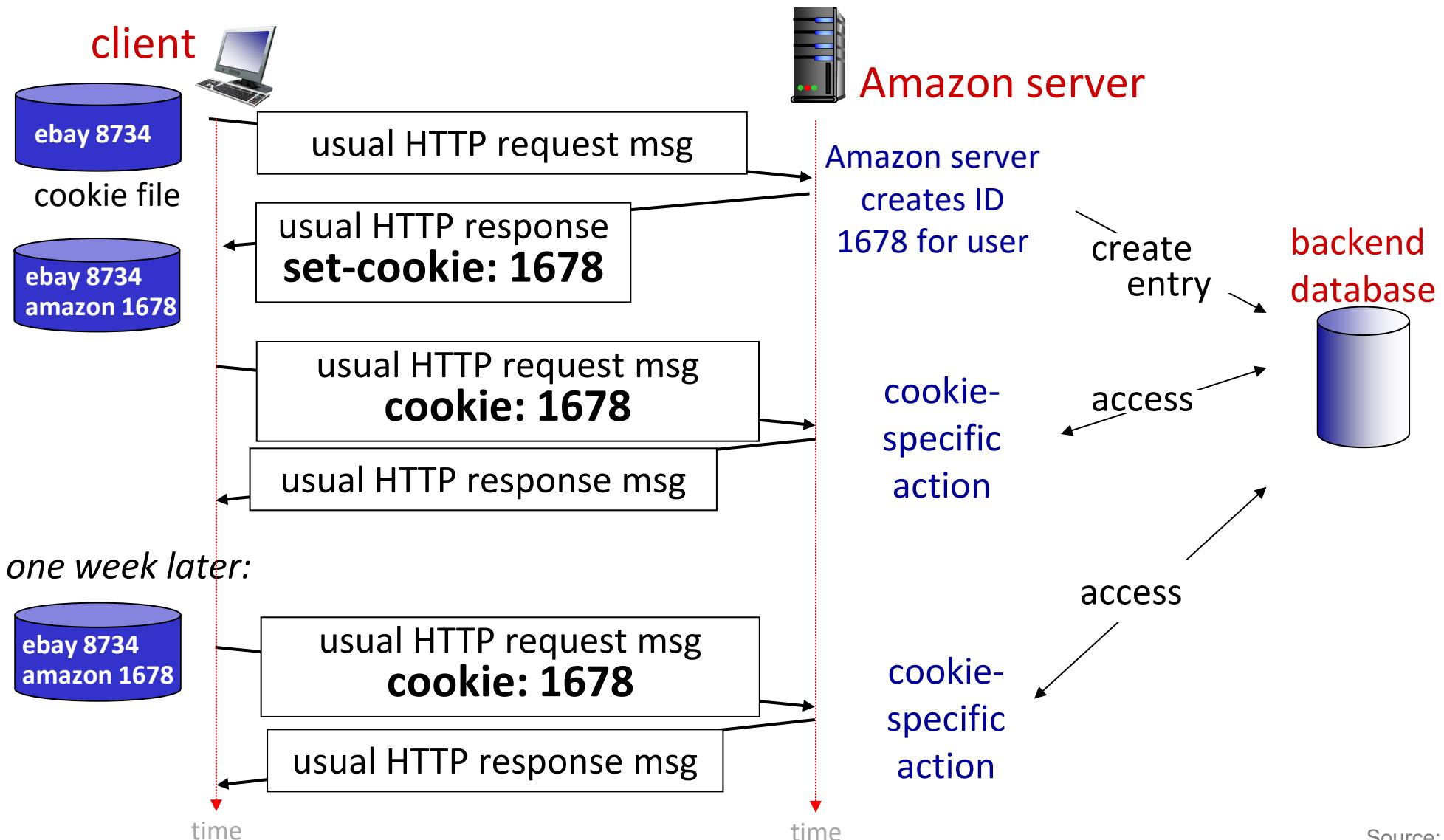
four components:

- 1) cookie header line of HTTP *response* message (from server)
- 2) cookie header line in next HTTP *request* message (from client)
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

- Susan uses browser on laptop, visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - unique ID (aka "cookie")
 - entry in backend database for ID
 - subsequent HTTP requests from Susan to this site will contain cookie ID value, allowing site to "identify" Susan

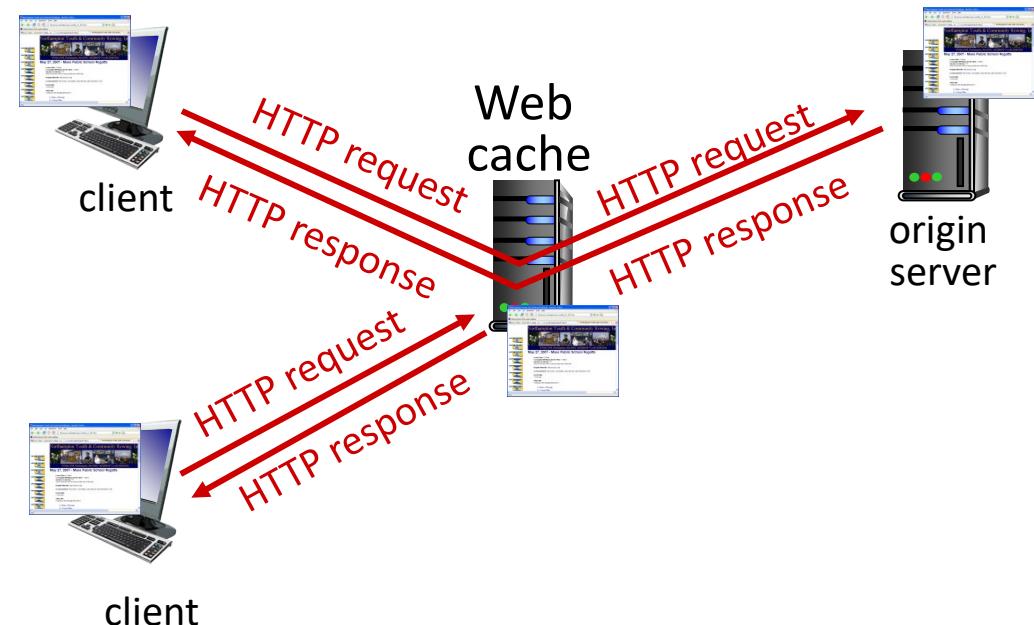
Maintaining user/server state: cookies



Web caches

Goal: satisfy client requests without involving origin server

- user configures browser to point to a (local) *Web cache*
- browser sends all HTTP requests to cache
 - *if* object in cache: cache returns object to client
 - *else* cache requests object from origin server, caches received object, then returns object to client



Web caches (aka proxy servers)

- Web cache acts as both client and server
 - server for original requesting client
 - client to origin server
- server tells cache about object's allowable caching in response header:

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

Why Web caching?

- reduce response time for client request
 - cache is closer to client
- reduce traffic on an institution's access link
- Internet is dense with caches
 - enables “poor” content providers to more effectively deliver content

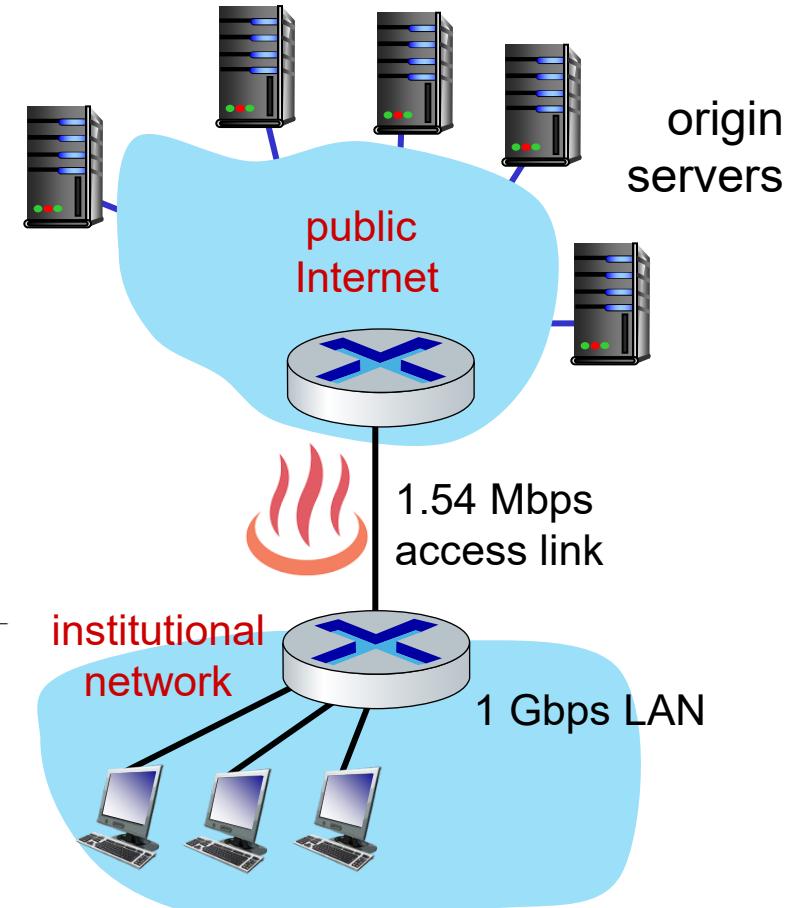
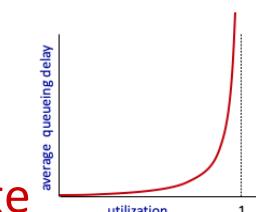
Caching example

Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
 - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = **.97** *problem: large queueing delays at high utilization!*
- LAN utilization: .0015
- end-end delay = Internet delay +
access link delay + LAN delay
= 2 sec + **minutes** + microsecs



HTTP/2

Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (**head-of-line (HOL) blocking**) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

HTTP/2

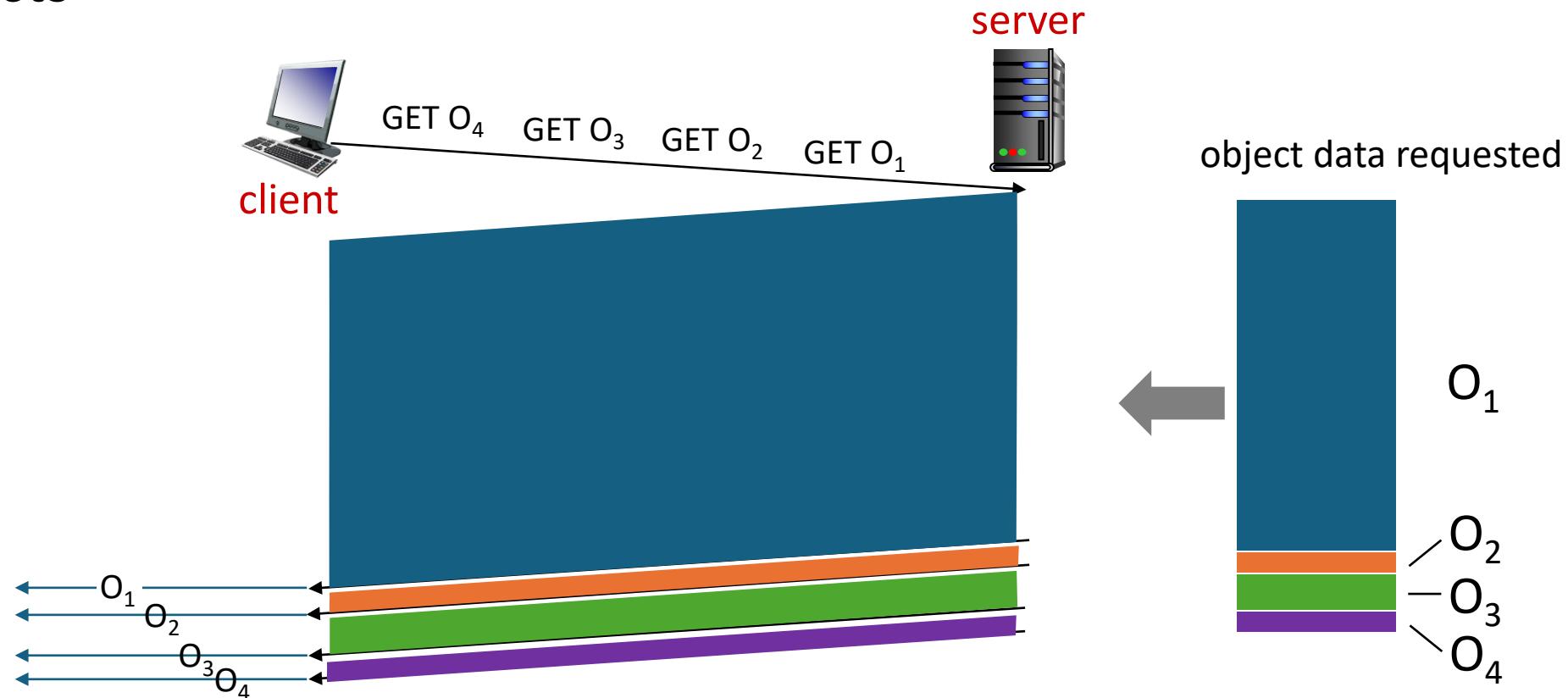
Key goal: decreased delay in multi-object HTTP requests

HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *push* unrequested objects to client
- divide objects into frames, schedule frames to mitigate HOL blocking

HTTP/2: mitigating HOL blocking

HTTP 1.1: client requests 1 large object (e.g., video file) and 3 smaller objects

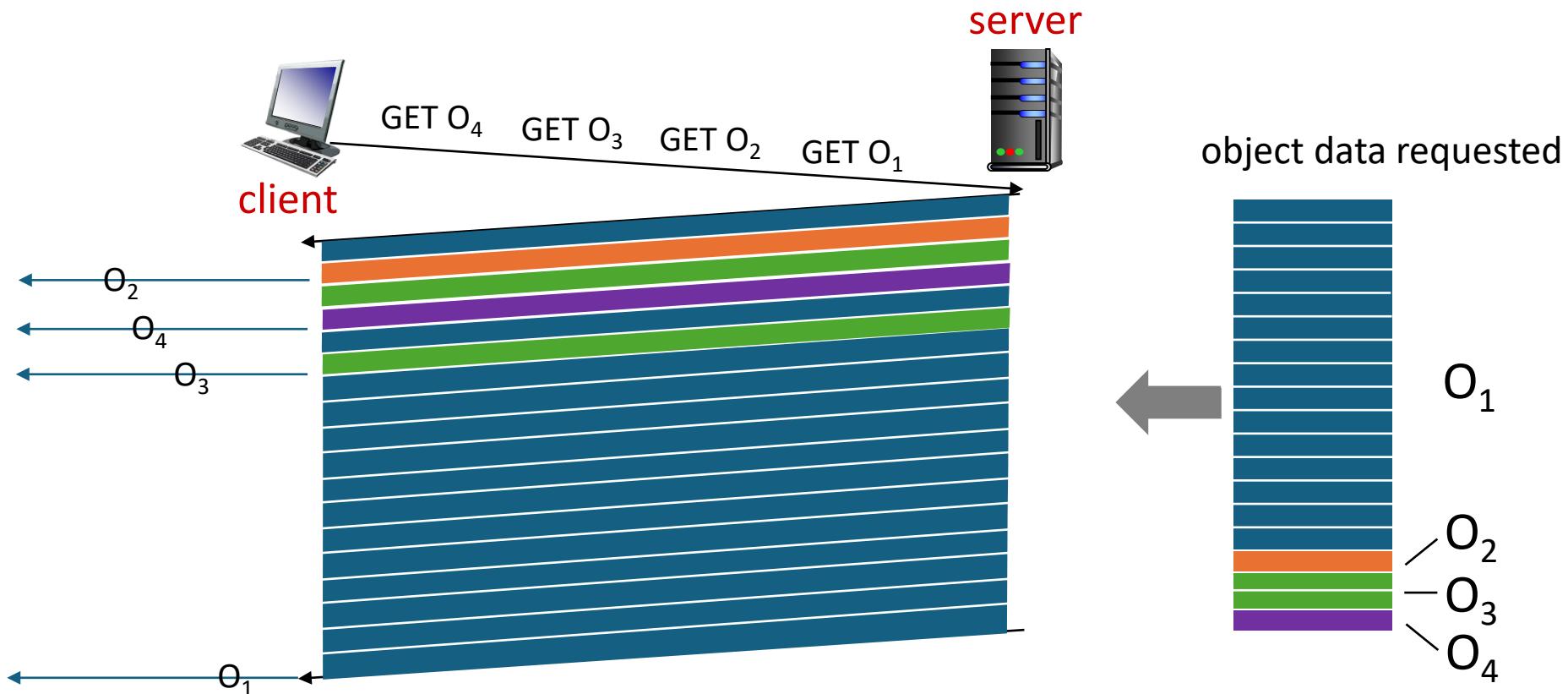


objects delivered in order requested: O₂, O₃, O₄ wait behind O₁

Source: Kurose & Ross

HTTP/2: mitigating HOL blocking

HTTP/2: objects divided into frames, frame transmission interleaved



O₂, O₃, O₄ delivered quickly, O₁ slightly delayed

HTML- HyperText Markup Language (1)

```
<html>
<head><title> AMALGAMATED WIDGET, INC. </title> </head>
<body><h1> Welcome to AWI's Home Page</h1>
 <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's </b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax.</p>
<hr>
<h2> Product information </h2>
<ul>
  <li> <a href="http://widget.com/products/big"> Big widgets </a>
  <li> <a href="http://widget.com/products/little"> Little widgets </a>
</ul>
<h2> Telephone numbers</h2>
<ul>
  <li> By telephone: 1-800-WIDGETS
  <li> By fax: 1-415-765-4321
</ul>
</body>
</html>
```

(a)

Welcome to AWI's Home Page



We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope you will find all the information you need here.

Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.

Product Information

- [Big widgets](http://widget.com/products/big)
- [Little widgets](http://widget.com/products/little)

Telephone numbers

- 1-800-WIDGETS
- 1-415-765-4321

(b)

(a) The HTML for a sample Web page (b) The formatted page

HTML (2)

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h <i>n</i> > ... </h <i>n</i> >	Delimits a level <i>n</i> heading
 ... 	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
 ... 	Brackets an unordered (bulleted) list
 ... 	Brackets a numbered list
	Starts a list item (there is no)
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a Horizontal rule
	Displays an image here
 ... 	Defines a hyperlink

A selection of common HTML tags. some can have additional parameters.