Computer Networks

CSE 303 and 304

The Interface Between the Process and the Computer Network

- The two processes in each pair sending messages to each other.
- Any message sent from one process to another must go through the underlying network.
- A process sends messages into, and receives messages from, the network through a software interface called a **socket**.
- When a process wants to send a message to another process on another host, it shoves the message out its door (socket).

The Interface Between the Process and the Computer Network

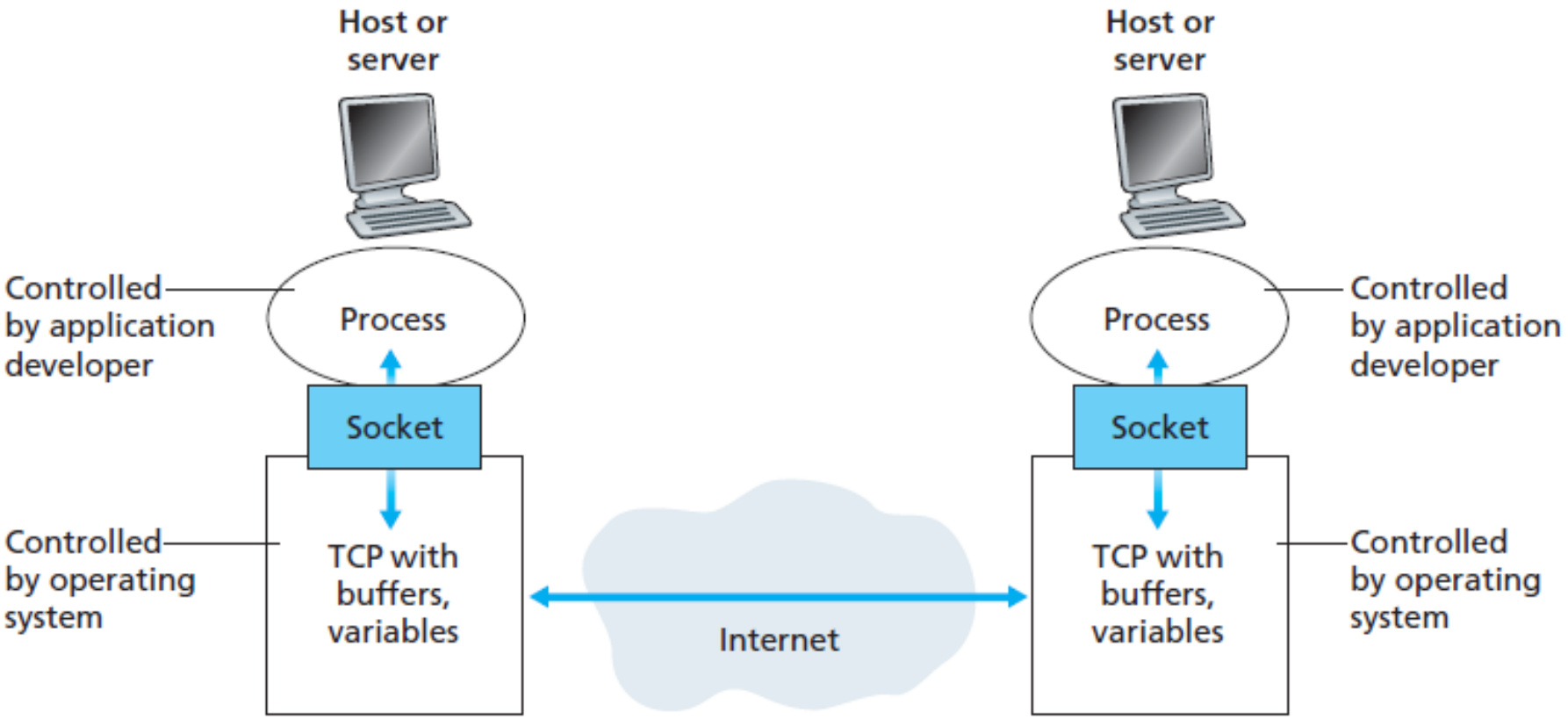

Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

The Interface Between the Process and the Computer Network

- A socket is the interface between the application layer and the transport layer within a host. It is also referred to as the **Application Programming Interface (API)** between the application and the network, since the socket is the programming interface with which network applications are built.
- The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket.
- The only control that the application developer has on the transport-layer side is
  - (1) the choice of transport protocol and
  - (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes

Application-Layer Protocols

- An **application-layer protocol** defines how an application's processes, running on different end systems, pass messages to each other.
- In particular, an application-layer protocol defines:
  - The types of messages exchanged, for example, request messages and response messages.
  - The syntax of the various message types, such as the fields in the message and how the fields are delineated
  - The semantics of the fields, that is, the meaning of the information in the fields
  - Rules for determining when and how a process sends messages and responds to messages.

The Web and HTTP

- The **Hyper Text Transfer Protocol (HTTP)**, the Web's application-layer protocol, is at the heart of the Web.
- HTTP is implemented in two programs: a client program and a server program.
- The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages.

The Web and HTTP

- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients.
- When a user requests a Web page (for example, clicks on a hyperlink), the browser sends HTTP request messages for the objects in the page to the server.
- The server receives the requests and responds with HTTP response messages that contain the objects.

The Web and HTTP

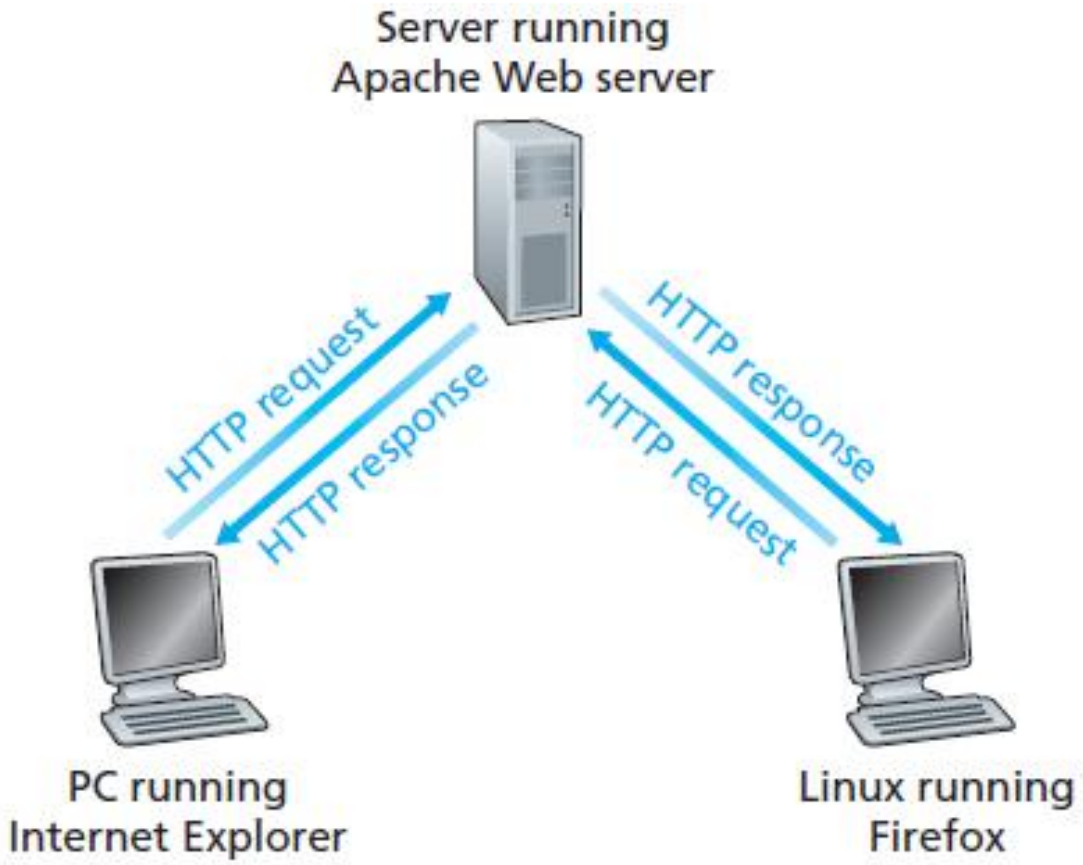

Figure 2.6 ♦ HTTP request-response behavior

The Web and HTTP

- HTTP uses TCP as its underlying transport protocol.
- The HTTP client first initiates a TCP connection with the server. Once the connection is established, the browser and the server processes access TCP through their socket interfaces.
- On the client side the socket interface is the door between the client process and the TCP connection; on the server side it is the door between the server process and the TCP connection.

HTTP with Non-Persistent Connections

- **Non-persistent connections:** each request/response pair be sent over a *separate* TCP connection.
- **Persistent connections:** all of the requests and their corresponding responses be sent over the *same* TCP connection.

Steps of non-persistent connections

- The HTTP client process initiates a TCP connection to the server Associated with the TCP connection, there will be a socket at the client and a socket at the server.
- The HTTP client sends an HTTP request message to the server via its socket.
- The HTTP server process receives the request message via its socket, retrieves the object, encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.

Steps of non-persistent connections

- The HTTP server process tells TCP to close the TCP connection. (But TCP doesn't actually terminate the connection until it knows for sure that the client has received the response message intact.)
- The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file.
- The first four steps are then repeated for each of the referenced objects.

Steps of non-persistent connections

- The steps above illustrate the use of non-persistent connections, where each TCP connection is closed after the server sends the object—the connection does not persist for other objects.
- Note that each TCP connection transports exactly one request message and one response message.

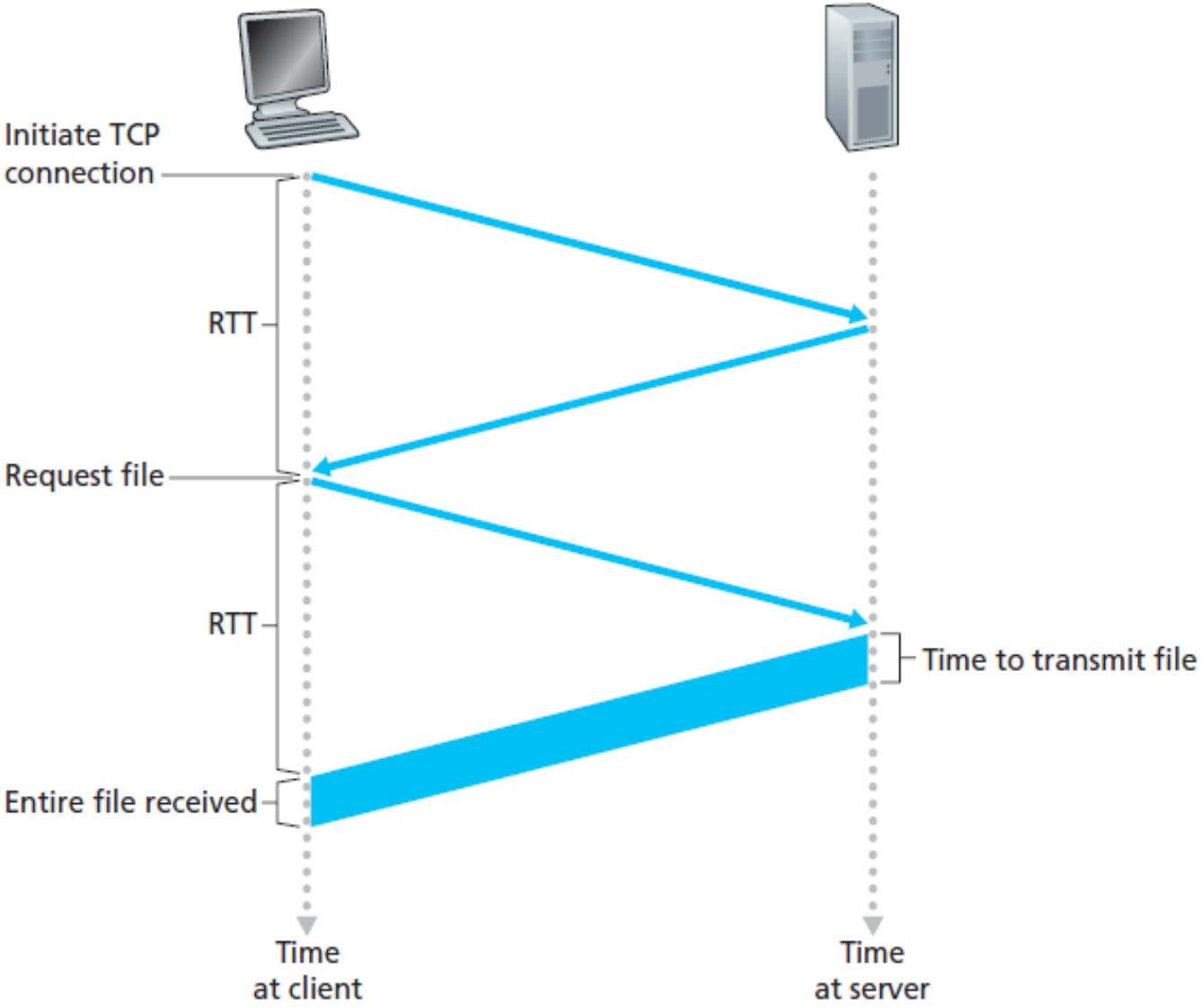

Figure 2.7 ♦ Back-of-the-envelope calculation for the time needed to request and receive an HTML file

The **round-trip time (RTT)**, which is the time it takes for a small packet to travel from client to server and then back to the client.

HTTP with Persistent Connections

- With persistent connections, the server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connection.
- In particular, an entire Web page (in the example above, the base HTML file and the 10 images) can be sent over a single persistent TCP connection.
- Moreover, multiple Web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection.
- These requests for objects can be made back-to-back, without waiting for replies to pending requests (pipelining). Typically, the HTTP server closes a connection when it isn't used for a certain time (a configurable timeout interval).

HTTP Message Format

- There are two types of HTTP messages:
  - Request messages and
  - Response messages

HTTP Request Message

Below we provide a typical HTTP request message:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP Request Message

- The message consists of five lines:
- The first line of an HTTP request message is called the **request line**;
  - The request line has three fields: the method field, the URL field, and the HTTP version field.
  - The method field can take on several different values, including GET, POST, HEAD, PUT, and DELETE. The great majority of HTTP request messages use the GET method.
  - The GET method is used when the browser requests an object, with the requested object identified in the URL field. In this example, the browser is requesting the object /somedir/page.html. The version is self-explanatory; in this example, the browser implements version HTTP/1.1.

HTTP Request Message

- The subsequent lines are called the **header lines**.
  - The header line **Host:** www.someschool.edu specifies the host on which the object resides.
  - The **Connection:** close, header line, the browser is telling the server that it doesn't want to bother with persistent connections; it wants the server to close the connection after sending the requested object.
  - 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/5.0, a Firefox browser.

HTTP Request Message

– 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 Accept-language: header is just one of many content negotiation headers available in HTTP.

General format of an HTTP request message

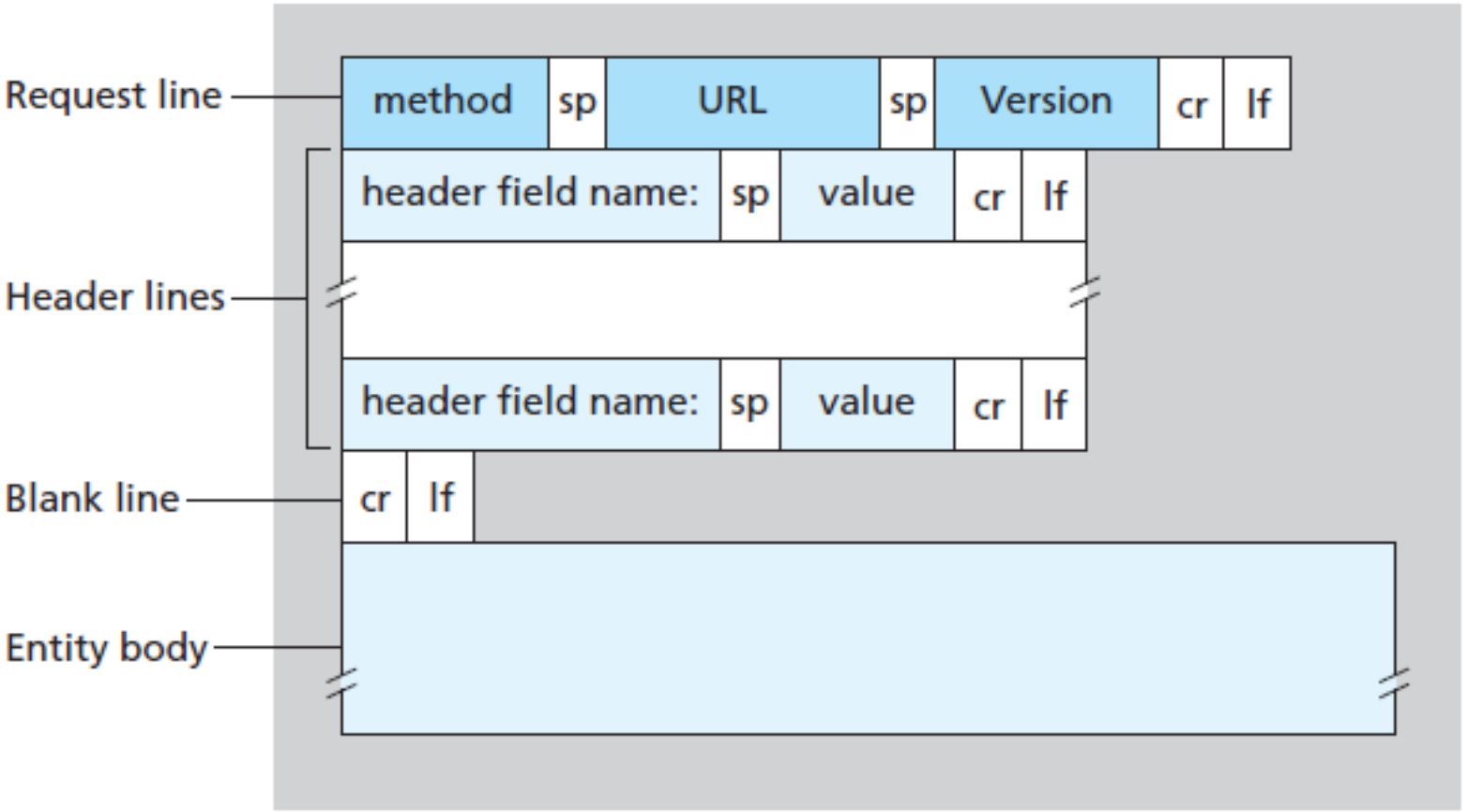

Figure 2.8 ♦ General format of an HTTP request message

HTTP Response Message

- Below we provide a typical HTTP response message:

  HTTP/1.1 200 OK
  Connection: close
  Date: Tue, 09 Aug 2011 15:44:04 GMT
  Server: Apache/2.2.3 (CentOS)
  Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
  Content-Length: 6821
  Content-Type: text/html
  (data data data data data ...)

HTTP Response Message

- It has three sections:
- An initial **status line**:
  - The **status line** has three fields: the protocol version field, a status code, and a corresponding status message. In this example, the status line indicates that the server is using HTTP/1.1 and that everything is OK.
- Six **header lines**:
  - The server uses the Connection: close header line to tell the client that it is going to close the TCP connection after sending the message.

HTTP Response Message

- The Date: header line indicates the time and date when the HTTP response was created and sent by the server.
- The Server: header line indicates that the message was generated by an Apache Web server; it is analogous to the User-agent: header line in the HTTP request message.
- The Last-Modified: header line indicates the time and date when the object was created or last modified.
- The Content-Length: header line indicates the number of bytes in the object being sent.
- The object type is officially indicated by the Content-Type: header and not by the file extension.

- Then the **entity body**:
  - The entity body is the meat of the message—it contains the requested object itself (represented by data data data data data ...).

General format of an HTTP response message

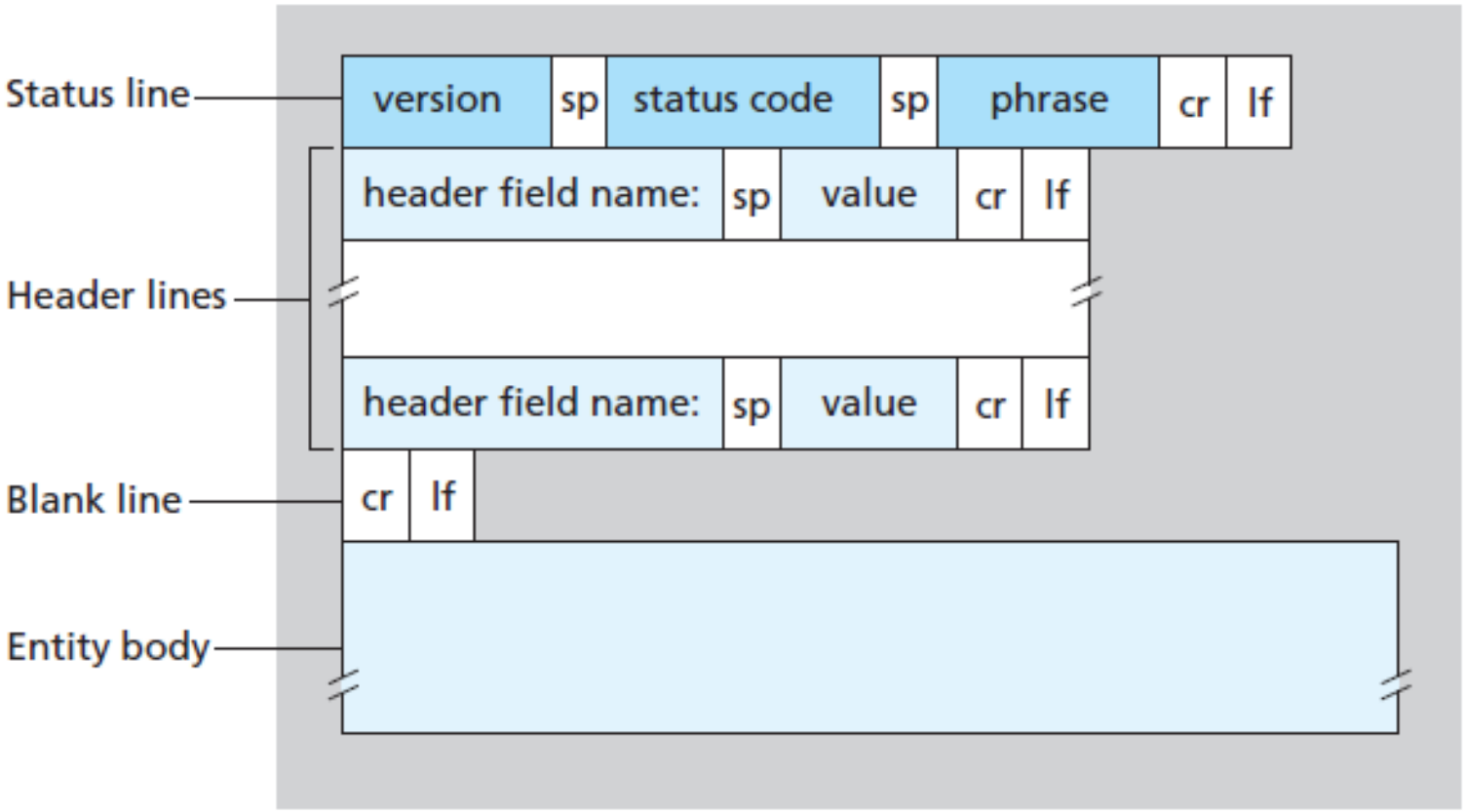

Figure 2.9 ♦ General format of an HTTP response message

HTTP Response Message

- Some common status codes and associated phrases include:
  - **200 OK:** Request succeeded and the information is returned in the response.
  - **301 Moved Permanently:** Requested object has been permanently moved;
  - **400 Bad Request:** This is a generic error code indicating that the request could not be understood by the server.
  - **404 Not Found:** The requested document does not exist on this server.
  - **505 HTTP Version Not Supported:** The requested HTTP protocol version is not supported by the server.

User-Server Interaction: Cookies

- To develop high-performance Web servers that can handle thousands of simultaneous TCP connections.
- However, it is often desirable for a Web site to identify users, either because-
  - the server wishes to restrict user access or because it wants to serve content as a function of the user identity.
- For these purposes, HTTP uses cookies. Cookies, defined in [RFC 6265], allow sites to keep track of users.
- Most major commercial Web sites use cookies today.

Cookies

- A cookie technology has four components:
  - (1) a cookie header line in the HTTP response message;
  - (2) a cookie header line in the HTTP request message;
  - (3) a cookie file kept on the user's end system and managed by the user's browser; and
  - (4) a back-end database at the Web site.

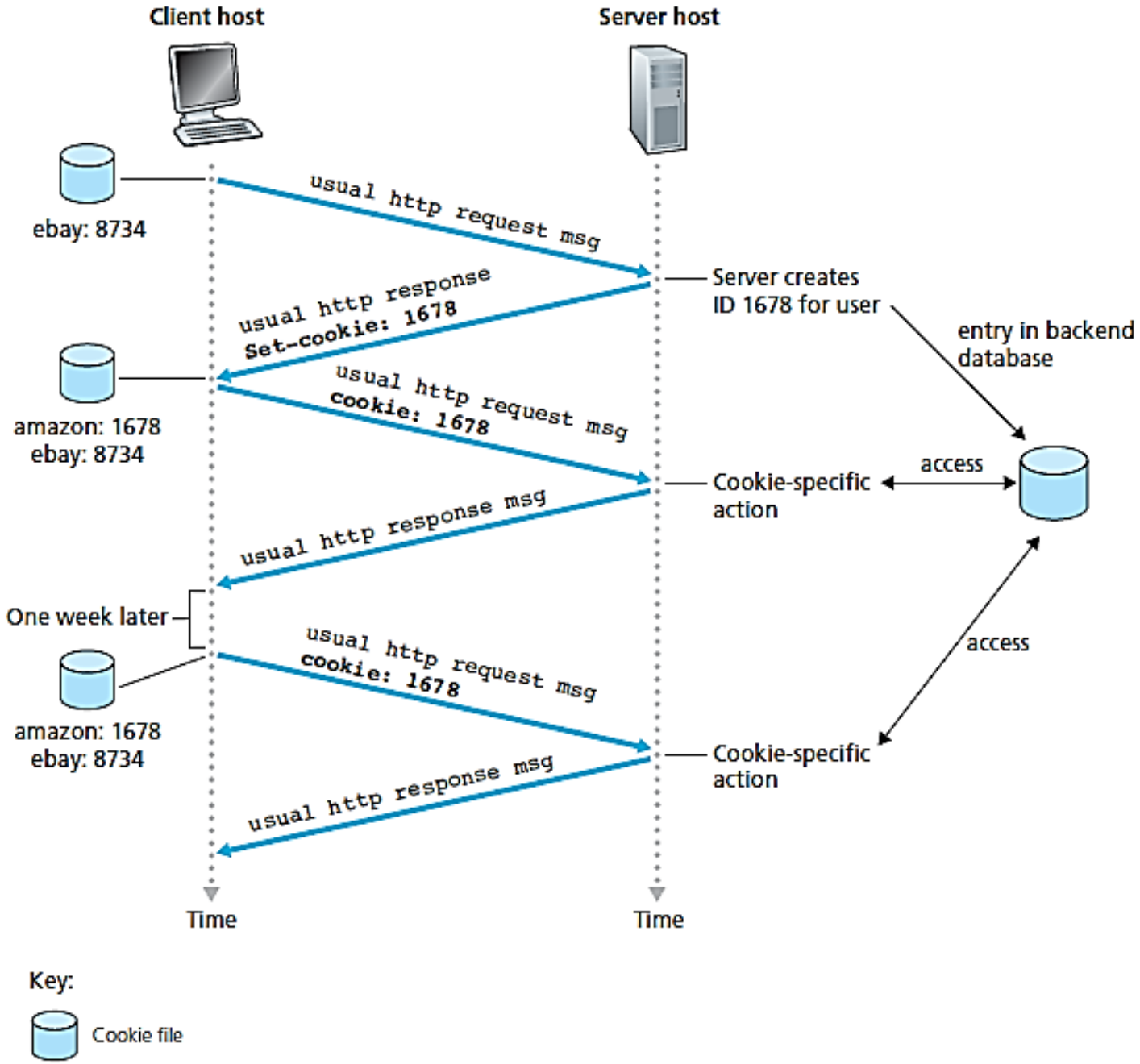

Figure 2.10 ♦ Keeping user state with cookies

Cookies

- When the request comes into the Amazon Web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number.
- The Amazon Web server then responds to Susan's browser, including in the HTTP response a Set-cookie: header, which contains the identification number. For example, the header line might be:

  Set-cookie: 1678

Cookies

- When Susan's browser receives the HTTP response message, it sees the Set cookie: header. The browser then appends a line to the special cookie file that it manages.
- This line includes the hostname of the server and the identification number in the Set-cookie: header.
- As Susan continues to browse the Amazon site, each time she requests a Web page, her browser consults her cookie file, extracts her identification number for this site, and puts a cookie header line that includes the identification number in the HTTP request.
- Specifically, each of her HTTP requests to the Amazon server includes the header line:

Cookie: 1678

Cookies

- The Amazon server is able to track Susan's activity at the Amazon site. Although the Amazon Web site does not necessarily know Susan's name, it knows exactly which pages user 1678 visited, in which order, and at what times! Amazon uses cookies to provide its shopping cart service

Cookies

- If Susan returns to Amazon's site, say, one week later, her browser will continue to put the header line Cookie: 1678 in the request messages.
- Amazon also recommends products to Susan based on Web pages she has visited at Amazon in the past. If Susan also registers herself with Amazon—providing full name, e-mail address, postal address, and credit card information—Amazon can then include this information in its database, thereby associating Susan's name with her identification number (and all of the pages she has visited at the site in the past!).
- This is how Amazon and other e-commerce sites provide "one-click shopping".

Web Caching

- A **Web cache**—also called a **proxy server**—is a network entity that satisfies HTTP requests on the behalf of an origin Web server.
- The Web cache has its own disk storage and keeps copies of recently requested objects in this storage.
- A user's browser can be configured so that all of the user's HTTP requests are first directed to the Web cache.
- Once a browser is configured, each browser request for an object is first directed to the Web cache.

Web Caching

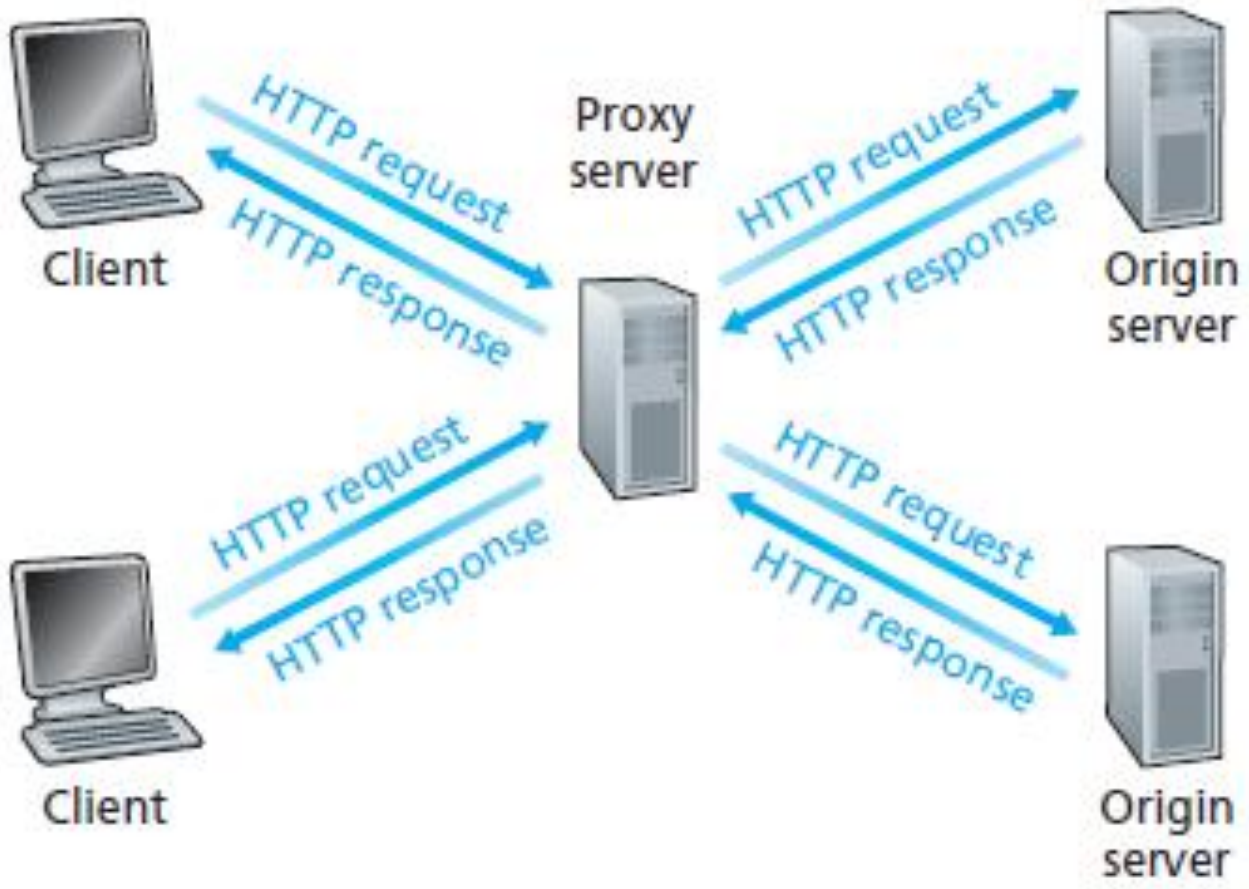

Figure 2.11 ♦ Clients requesting objects through a Web cache

Web Caching

- As an example, suppose a browser is requesting the objec http://www.someschool.edu/campus.gif.
- Here is what happens:
  - The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
  - The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.

Web Caching

- If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.someschool.edu. The Web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the Web cache.
- When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser

Web Caching

- Note that a cache is both a server and a client at the same time.
  - When it receives requests from and sends responses to a browser, it is a server.
  - When it sends requests to and receives responses from an origin server, it is a client.

Web Caching

- Web caching has seen deployment in the Internet for two reasons.
  - First, a Web cache can substantially reduce the response time for a client request, particularly if the bottleneck bandwidth between the client and the origin server is much less than the bottleneck bandwidth between the client and the cache.
  - Web caches can substantially reduce Web traffic in the Internet as a whole, thereby improving performance for all applications.

Domain Name System (DNS)

- There are two ways to identify a host—
  - by a hostname and
  - by an IP address.
- People prefer the more mnemonic hostname identifier, while routers prefer fixed-length, hierarchically structured IP addresses.
- In order to reconcile these preferences, we need a directory service that translates hostnames to IP addresses.
- This is the main task of the Internet's **domain name system (DNS)**.

Domain Name System (DNS)

- The DNS is
  - (1) a distributed database implemented in a hierarchy of **DNS servers**, and
  - (2) an application-layer protocol that allows hosts to query the distributed database.
- DNS is commonly employed by other application-layer protocols—including HTTP, SMTP, and FTP—to translate user-supplied hostnames to IP addresses.

Domain Name System (DNS)

- what happens when a browser running on some user's host, requests the URL www.someschool.edu/ index.html.
  - 1. The same user machine runs the client side of the DNS application.
  - 2. The browser extracts the hostname, www.someschool.edu, from the URL and passes the hostname to the client side of the DNS application.
  - 3. The DNS client sends a query containing the hostname to a DNS server.
  - 4. The DNS client eventually receives a reply, which includes the IP address for the hostname.
  - 5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.

Domain Name System (DNS)

- DNS provides a few other important services in addition to translating hostnames to IP addresses:
- **Host aliasing:**
  - a hostname such as **relay1.west-coast.enterprise.com** could have, say, two aliases such as **enterprise.com** and **www.enterprise.com.**
  - In this case, the hostname relay1.westcoast. enterprise.com is said to be a **canonical hostname**.

Domain Name System (DNS)

- **Mail server aliasing:**
  - Bob's e-mail address might be as simple as bob@hotmail.com.
  - However, the hostname of the Hotmail mail server is more complicated and much less mnemonic than simply hotmail.com (for example, the canonical hostname might be something like relay1.west-coast.hotmail.com).
  - A company's Web server and mail server can both be called enterprise.com.

Domain Name System (DNS)

- **Load distribution.**
  - DNS is also used to perform load distribution among replicated servers, such as replicated Web servers.
  - Busy sites, such as cnn.com, are replicated over multiple servers, with each server running on a different end system and each having a different IP address.
  - For replicated Web servers, a *set* of IP addresses is thus associated with one canonical hostname. The DNS database contains this set of IP addresses.
  - When clients make a DNS query for a name mapped to a set of addresses, the server responds with the entire set of IP addresses, but rotates the ordering of the addresses within each reply.

Portion of the hierarchy of DNS servers

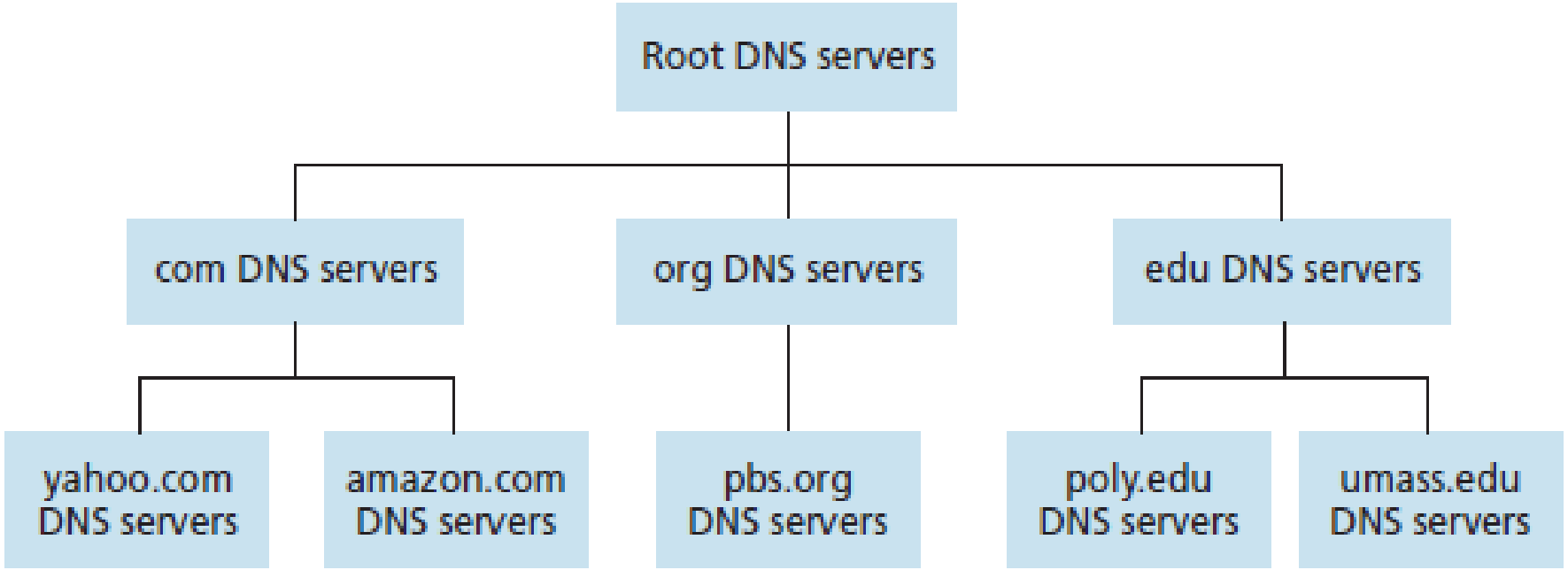

Figure 2.19 ♦ Portion of the hierarchy of DNS servers

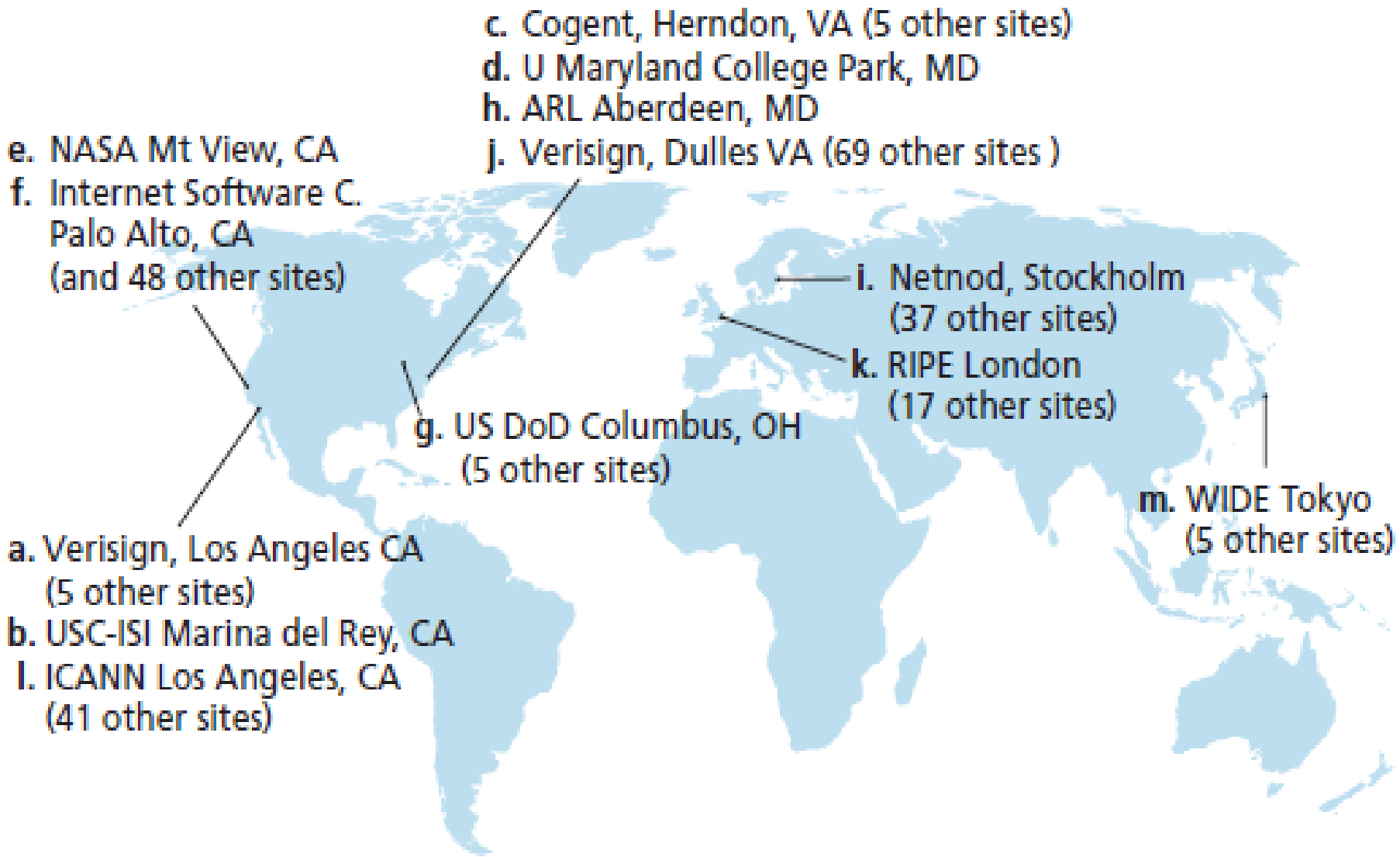

Figure 2.20 ♦ DNS root servers in 2012 (name, organization, location)

classes of DNS servers

- Three classes of DNS servers:
- **Root DNS servers.**
  - In the Internet there are 13 root DNS servers (labeled A through M), most of which are located in North America.
  - a list of the current root DNS servers
  - Each of the 13 root DNS servers as if it were a single server, each “server” is actually a network of replicated servers, for both security and reliability purposes.

classes of DNS servers

- **Top-level domain (TLD) servers.**
  - These servers are responsible for top-level domains such as com, org, net, edu, and gov, and all of the country top-level domains such as uk, fr, ca, and jp.
- **Authoritative DNS servers.**
  - Every organization with publicly accessible hosts (such as Web servers and mail servers) on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses. An organization's authoritative DNS server houses these DNS records.

End