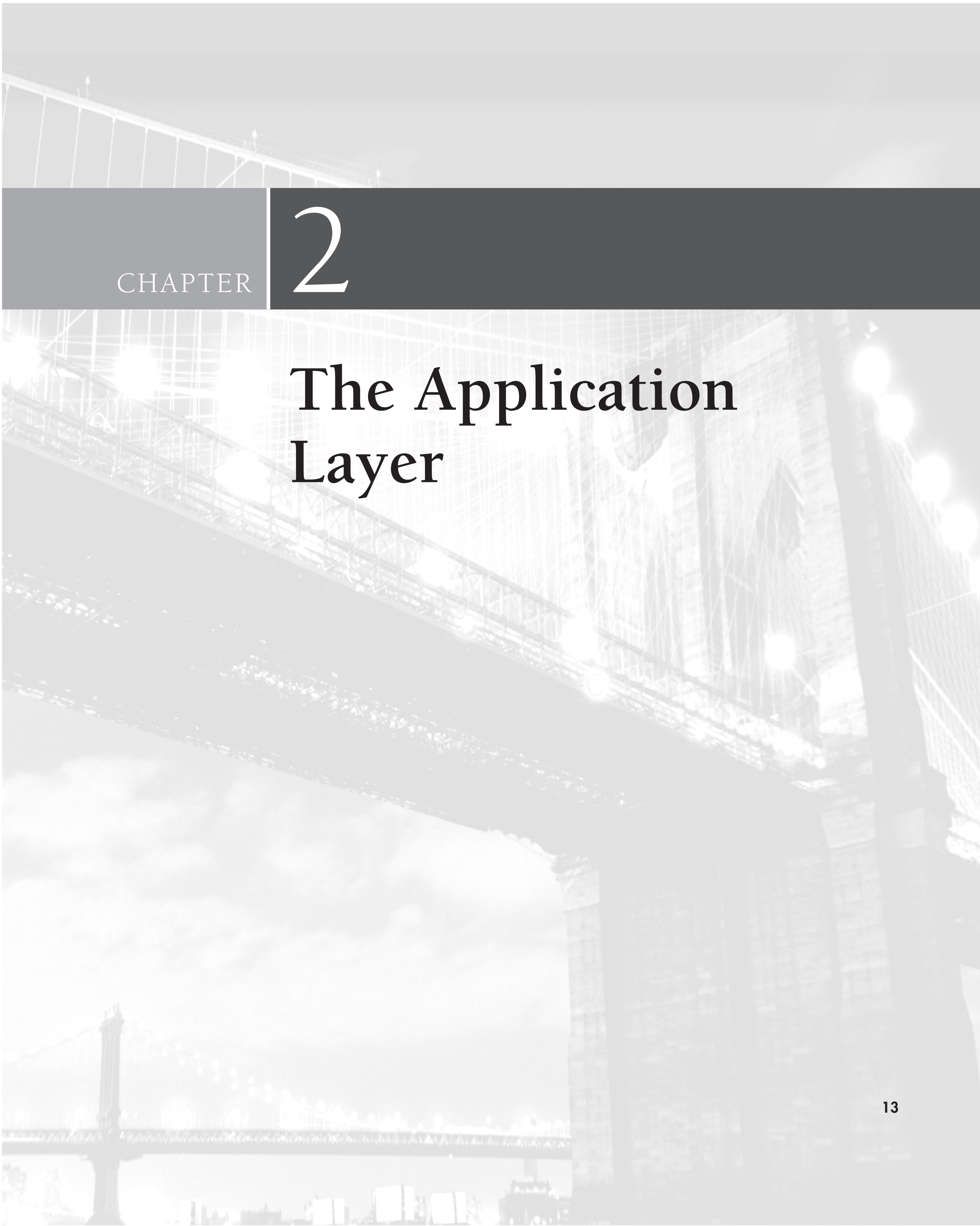


CHAPTER 2

The Application Layer

Most Important Ideas and Concepts from Chapter 2

- **Application-layer protocol.** In Chapter 1 we noted that "*A* **protocol** *defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event*." In chapter 2, we have seen how processes send and receive messages in an application-layer protocol. As a review, identify the messages exchanged and actions taken by the following protocols: HTTP, FTP, DNS, SMTP.
- **Client/server versus peer-to peer.** These are the two approaches that we studied structuring a network application. In the client/server paradigm (see page 75 of the textbook), a client process requests a service by sending one or more messages to a server process. The server process implements a service by reading the client request, performing some action (for example, in the case of an HTTP server, finding a Web page), and sending one or more messages in reply (in the case of HTTP, returning the requested object). In a peer-to-peer approach, the two ends of the protocol are equals (as in a telephone call).
- **Two services provided by the Internet's transport layer: reliable, congestion-controlled data transfer (TCP), and unreliable data transfer (UDP).** These are the only services available to an Internet application to transfer data from one process to another remote process. The Internet transport layer does not provide a minimum guaranteed transfer rate, or a bound on the delay from source to destination.
- **HTTP: request/response interaction.** The HTTP protocol is a simple application-layer protocol. A client (Web browser) makes a request with a GET message, and a Web server provides a reply (see Figure 2.6 on page 89 in your textbook). This is a classical client/server approach. Since HTTP uses TCP to provide reliable transfer of the GET request from client-to-server, and the reply from server-to-client, a TCP connection must be set up. A TCP setup request is sent from the TCP in the client to the TCP in the server, with the TCP server replying to the TCP client. Following this exchange, the HTTP GET message can be sent over the TCP connection from client-to-server, and the reply received (see Figure 2.7 on page 92 in your textbook). With non-persistent HTTP, a new TCP connection must be set up each time the client wants to contact the server. With persistent HTTP, multiple HTTP GET messages can be sent over a single TCP connection, resulting in performance gains from not having to set up a new TCP for each of the HTTP requests beyond the first.
- **Caching.** Caching is the act of saving a local copy of a requested piece of information (for example, Web document or DNS translation pair) that is retrieved from a distant location, so that if the same piece of information is requested again, it can be retrieved from the local cache, rather than having to retrieve the information again from the distant location. Caching can improve performance by decreasing response time (since the local cache is closer to the requesting client) and avoiding

the use of scare resources (such as the 1.5 Mbps access link shown in Figures 2.11 on page 103 and 2.12 on page 104 in your textbook). Think about ways you use caching in your every day life—e.g., writing a phone number on a piece of paper and keeping it in your pocket, rather than looking it up again in a phone book.

- **DNS: core infrastructure implemented as an application-layer process.** The DNS is an application-layer protocol. The name-to-IP-address translation service is performed at the DNS servers, just as any application provides a service to a client via a server. But the DNS service is a very special *network* service—without it the network would be unable to function. Yet it is implemented in very much the same way as any other network application.
- **FTP: separate control and data.** Students often ask us why we include an "old" application such as FTP here. FTP is a nice example of a protocol that separates control and data messages. As shown in Figure 2.15 (on page 110 in your textbook), control and data messages are sent over separate TCP connections. This logical and physical separation of control and data (rather than mixing the two types of messages in one connection) helps to make the structure of such an application "cleaner."
- **TCP sockets: accept(), and the creation of a new socket.** A "tricky" thing about TCP sockets is that a new socket is created when a TCP server returns from an accept() system call. We call the socket on which the server waits when performing the accept() as a "welcoming socket." The socket returned from the accept() is used to communicate back to the client that connected to the server via the accept() (see Figure 2.28 on page 150 in your textbook).
- **UDP sockets: send and pray on the receiving side; datagrams from many senders on the receiving side.** Since UDP provides an unreliable data transfer service, a sender that sends a datagram via a UDP socket has no idea if the datagram is received by the receiver (unless the receiver is programmed to send back a datagram that acknowledges that the original datagram was received). On the receiving side, datagrams from many different senders can be received on the same socket.
- **Pull versus push.** How does one application process get data to or from another application process? In a pull system (such as the Web), the data receiver must explicitly request ("pull") the information. In a push system, the data holder sends the information to the receiver without the receiver's explicitly asking for the data (as in SMTP, when an email is "pushed" from sender to receiver).
- **Locating information in P2P systems.** We identified three ways to locate information in a P2P system: query flooding, directory systems, and hybrid systems. All existing P2P systems use one of these approaches.

Review Questions

This section provides additional study questions. Answers to each question are provided in the next section.

1. **Client-server, P2P, or Hybrid?** Section 2.1.1 in your textbook discusses three application architectures: client-server, P2P, and a hybrid of the two. Classify each of the scenarios below as client-server, P2P, or hybrid, and explain your answer briefly. Answering these questions may require some Web surfing.
   a. EBay
   b. Skype
   c. BitTorrent
   d. Telnet
   e. DNS
2. **Services provided by the Internet transport protocols.** Indicate whether TCP or UDP (or both or neither) provide the following services to applications:
   a. Reliable data transfer between processes.
   b. Minimum data transmission rate between processes.
   c. Congestion-controlled data transfer between processes.
   d. A guarantee that data will be delivered within a specified amount of time.
   e. Preserve application-level message boundaries. That is, when a sender sends a group of bytes into a socket via a single send operation, that group of bytes will be delivered as a group in a single receive operation at the receiving application.
   f. Guaranteed in-order delivery of data to the receiver.
3. **Fast transactions.** Suppose you want to do a transaction from a remote client to a server as fast as possible, would you use UDP or TCP?
4. **Reliable data transfer with UDP.** Suppose you use UDP to do a transaction from a remote client to a server. UDP provides no reliability, but you want your transaction request to be sent reliably. How could you do is?
5. **Timeliness or in-order delivery.** Suppose that data is being output at a sensor at 1 sample per second. It is important for the receiver to have the most recent value of the sensor's reading, rather than all values (for example, it is better for the receiver to get a current value, rather than an outdated value). Would you use TCP or UDP to send this sensor data? Why?
6. **HTTP basics.** Consider the following string of ASCII characters that were captured by Ethereal when the browser sent an HTTP GET message (this is the

actual content of an HTTP GET message). The characters <cr><lf> are carriage return and line feed characters (that is, the italicized character string <cr> in the text below represents the single carriage-return character that was contained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answers.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml,application/xml,application/xhtml+xml,text
/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5..Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

a. What is the URL of the document requested by the browser? Make sure you give the hostname and the file name parts of the URL.
b. What version of HTTP is the browser running?
c. Is a Netscape or an Internet Explorer browser making the request?
d. Is the browser requesting a non-persistent or a persistent connection?
e. What is the IP address of the computer on which the browser is running?

7. **More HTTP basics.** The text below shows the reply sent from the server in response to HTTPP GET message in Question 6. Answer the following questions, indicating where in the message below you find the answers.

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2006
12:39:45GMT..Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec 2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>Keep-
Alive: timeout=max=100<cr><lf>Connection: Keep-
Alive<cr><lf>Content-Type: text/html; charset=ISO-8859-
1<cr><lf><cr><lf><!doctype html public "-//w3c//dtd html
4.0 transitional//en"><lf><html><lf><head><lf>  <meta
http-equiv="Content-Type" content="text/html;
charset=iso-8859-1"><lf> <meta name="GENERATOR"
content="Mozilla/4.79 [en] (Windows NT 5.0; U)
Netscape]"><lf>  <title>CMPSCI 453 / 591 / NTU-ST550A
Spring 2005 homepage</title><lf></head><lf> <much more
document text following here (not shown)>
```

a. Was the server able to find the document successfully or not?

b. At what time was the document reply provided?

c. When was the document last modified?

d. How many bytes are there in the document being returned?

e. What are the first 5 bytes of the document being returned?

f. Did the server agree to a persistent connection?

8. **HTTP Performance.** Here, we consider the performance of HTTP, comparing non-persistent HTTP with persistent HTTP. Suppose the page your browser wants to download is 100K bits long, and contains 10 embedded images (with file names img01.jpg, img02.jpg, … img10.jpg), each of which is also 100K bits long. The page and the 10 images are stored on the same server, which has a 300 msec roundtrip time (RTT) from your browser. We will abstract the network path between your browser and the Web server as a 100 Mbps link. You can assume that the time it takes to transmit a GET message into the link is zero, but you should account for the time it takes to transmit the base file and the embedded objects into the "link." This means that the server-to-client "link" has both a 150 msec one-way propagation delay, as well as a transmission delay associated with it. (Review page 39 in the textbook if you are uncertain about the difference between transmission delay and propagation delay.) In your answer, be sure to account for the time needed to set up a TCP connection (1 RTT).

a. Assuming non-persistent HTTP (and assuming no parallel connections are open between the browser and the server), how long is the *response time*—the time from when the user requests the URL to the time when the page and its embedded images are displayed? Be sure to describe the various components that contribute to this delay.

b. Again, assume non-persistent HTTP, but now assume that the browser can open as many parallel TCP connections to the server as it wants. What is the response time in this case?

c. Now assume persistent HTTP (HTTP1.1). What is the response time, assuming no pipelining?

d. Now suppose persistent HTTP with pipelining is used. What is the response time?

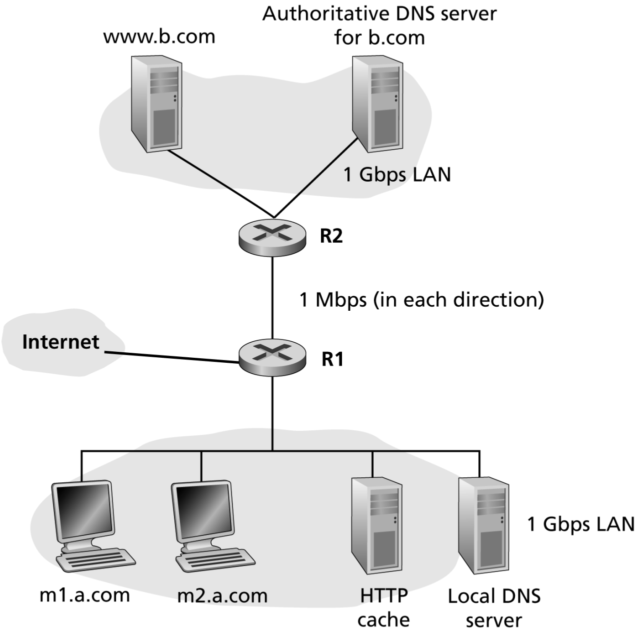
9. **Caching and delays.** Consider the networks shown in the figure below. There are two user machines—m1.a.com and m2.a.com in the network a.com. Suppose the user at m1.a.com types in the URL www.b.com/bigfile.htm into a browser to retrieve a 1Gbit (1000 Mbit) file from www.b.com.

a. List the sequence of DNS and HTTP messages sent/received from/by m1.a.com, as well as any other messages that leave/enter the a.com network that are not directly sent/received by m1.a.com from the point that the URL is entered into the browser until the file is completely received. Indicate the source and destination of each message. You can assume that every HTTP request by m1.a.com is first directed to the HTTP cache in a.com, that the cache is initially empty, and that all DNS requests are iterated queries.

b. How long does it take to accomplish the steps you outlined in your answer to the previous question regarding the m1a.com HTTP and DNS messages. Explain how you arrived at your answer. In answering this question, you can assume the following:
   - The packets containing DNS commands and HTTP commands such as GET are very small compared to the size of the file. Therefore, their transmission times (but not their propagation times) can be neglected.
   - Propagation delays within the local area networks (LANs) are small enough to be ignored. The propagation from router R1 to router R2 is 100 msec.
   - The one-way propagation delay from anywhere in a.com to any other site in the Internet (except b.com) is 500 msec.

c. Now assume that machine m2.a.com makes a request to the same URL that m1.a.com requested. List the sequence of DNS and HTTP messages sent/received from/by m2.a.com as well as other messages that leave/enter the a.com network that are not directly sent/received by m2.a.com from the point that the URL is entered into the browser until the file is completely received. Indicate the source and destination of each message. (Hint: be sure to consider caching.)

d. Now suppose there is no HTTP cache in network a.com. What is the maximum rate at which machines in a.com can make requests for the file www.b.com/bigfile.htm while keeping the time from when a request is made to when it is satisfied non-infinite in the long run?



10. **Persistent versus non-persistent TCP connections.** Suppose within your Web browser you click on a link to obtain a Web page. Suppose the IP address for the associated URL is cached in your local host, so that a DNS lookup is not necessary. Denote RTT as the roundtrip time between the local host and the server containing the Web page. Assume the Web page consists of a base HTML file and three small images. Assume the transmission times for all of the objects are negligible in comparison with the RTT. How much time elapses (in terms of RTTs) from when the user clicks on the link until the client receives the entire Web page with each of the following?

   a. Non-persistent HTTP with no parallel connections
   b. Non-persistent HTTP with up to five parallel connections
   c. Persistent HTTP with pipelining

11. **In-band versus out-of band control.** What does it mean when we say that control messages are “in-band”? What does it mean when we say that control messages are “out-of-band”? Give an example of a protocol that has in-band control messages and an example of a protocol that has out-of-band control messages.

12. **Networking over a slow, short link.** Consider a short (10 meter) link, over which a sender can transmit at a rate of 100 bits/sec in both directions. Suppose that packets containing data are 100Kbits long, and packets containing only control (for example, ACK or handshaking) are 100 bits long. Assume that N parallel connections each get 1/N of the link bandwidth.

    Now consider the HTTP protocol, and suppose that each downloaded object is 100Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Explain your answer.

13. **Who is sending this packet?** Is the following statement true or false? With UDP sockets, a server can easily determine the IP address of the client, from the data returned via a socket read. Answer the same question (true or false?) for TCP sockets. Briefly explain your answers.

14. **Caching.** Describe two ways in which caching is used in Web access. Describe one way in which caching is used in DNS.

15. **Mail.** What is the difference between the MAIL FROM: in SMTP and From: in the mail message itself?

16. **Push versus pull.** Consider the following forms of non-Internet data distribution: FM radio, broadcast TV, and newspapers. Are these "push" or "pull" systems? Explain. Is the traditional Web browser/server a "push" or "pull" system? Explain. Are there any non-Internet "pull" systems in your life?

17. **Query flooding in P2P networks.** Here, we explore the reverse-path routing of the QueryHit messages in Gnutella. Suppose that Alice issues a Query message. Furthermore, suppose that Bob receives the Query message (which may have been forwarded by several intermediate peers) and has a file that matches the query.

    a. Recall that when a peer has a matching file, it sends a QueryHit message along the reverse path of the corresponding Query message. An alternative design would be for Bob to establish a direct TCP connection with Alice and send the QueryHit message over this connection. What are the advantages and disadvantages of such an alternative design?

    b. In the Gnutella protocol, when the peer Alice generates a Query message, it inserts a unique ID in the message's MessageID field. When the peer Bob has a match, it generates a QueryHit message using the same MessageID as the Query message. Describe how peers can use the MessageID field and local routing tables to accomplish reverse-path routing.

    c. An alternative approach, which does not use message identifiers, is as follows. When a query message reaches a peer, before forwarding the message, the peer augments the query message with its IP address. Describe how peers can use this mechanism to accomplish reverse-path routing.

18. **TCP sockets: accept().** What is the purpose of the connection-oriented welcoming socket, which the server uses to perform an accept()? Once the accept() is done, does the server use the welcoming socket to communicate back to the client? Explain your answer.

19. **How many port numbers are in use?** Consider incoming TCP and UDP segments arriving at a server, and suppose that (for example, using Ethereal), we see that 150 different destination port numbers are being used. The server acts only as a server (that is, in the socket sense, it does not initiate communication with any other computers as a client; it only responds to incoming segments). Can the number of sockets in use at the server be (a) larger than, (b) equal to, or (c) less than 150? Explain your answer.

20. **Socket programming.** The skeleton of TCPServer.java is given below. This server receives a sentence from a client on port 6789, capitalizes the sentence, and sends the sentence back to the client. Answer the following questions without looking at the code in the textbook:

    a. Provide the one line of code that belongs at location (a).

    b. Provide the one line of code that belongs at location (b).

```
import java.io.*;
import java.net.*;
class TCPServer {
  public static void main(String argv[]) throws Exception
    {
      String clientSentence;
      String capitalizedSentence;
      // (a) Insert line of code
          while(true) {
    // (b) Insert line of code
          BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
    clientSentence =
    inFromClient.readLine();
    capitalizedSentence =
    clientSentence.toUpperCase() + '\n';
    outToClient.writeBytes(capitalizedSentence);
              }
    }
}
```

Answers to Review Questions

1. a. EBay is pure client-server application architecture. Ebay is implemented as a Web server (more accurately, a farm of Web servers) that responds to Web client (browser) requests using HTTP.

   b. When two Skype clients talk to each other, they do so as peers. However, in order to locate a peer, a Skype client will first contact a directory server in a client-server manner. Therefore, Skype has a hybrid application architecture.

   c. BitTorrent is a pure P2P application architecture. It is interesting because it will concurrently download different pieces of a file from different peers.

   d. Telnet is a pure client-server application architecture. The client is the process that contacts the Telnet server (at port 23 to allow remote login on the remote machine where the Telnet server process is running).

   e. DNS is a pure client-server application architecture. The DNS client is the process that sends the DNS REQUEST message (to port 53 at the DNS server); the server is the DNS server process that replies with a DNS REPLY message.

2 a. TCP provides a reliable byte-stream between client and server.

   b. Neither

   c. TCP

   d. Neither

   e. UDP preserves message boundaries, while TCP is byte-stream oriented, and does not preserve message boundaries.

   f. TCP will deliver bytes to the receiver in the order in which they were sent. UDP does not guarantee delivery of message data in the order in which they were sent.

3. You would use UDP. With UDP, the transaction can be completed in one roundtrip time (RTT)—the client sends the transaction request into a UDP socket, and the server sends the reply back to the client's UDP socket. With TCP, a minimum of two RTTs are needed—one to set-up the TCP connection, and another for the client to send the request, and for the server to send back the reply.

4. You would build reliability into your application. This could be done, for example, by having the client re-transmit its request if it doesn't hear back from the server within a certain amount of time. We will cover techniques to provide reliable data transfer in Chapter 3.

5. You would use UDP. With TCP, data (even old data) will be sent again and again until it is received correctly. Moreover, since data is passed up in order

by the TCP receiver, newer data will not be delivered to the receiving application until all old data has been delivered to the application. With UDP, if data is lost, newer data will eventually be sent and received, without waiting for the lost data to be recovered.

6. a. The document request was http://gaia.cs.umass.edu/cs453/index.html. The `Host`: field indicates the server's name and `/cs453/index.html` indicates the file name.

   b. The browser is running HTTP version 1.1, as indicated just before the first `<cr><lf>` pair.

   c. A Netscape browser is making the request. The `User-agent:` field indicates "Mozilla," which is a nickname for Netscape's browser.

   d. The browser is requesting a persistent connection, as indicated by the `Connection:keep-alive`.

   e. This is a trick question. This information is not contained in an HTTP message anywhere. So there is no way to tell this from looking at the exchange of HTTP messages alone. One would need information from the IP datagrams (that carried the TCP segment that carried the HTTP GET request) to answer this question.

7. a. The status code of `200` and the phrase `OK` indicate that the server was able to locate the document successfully.

   b. The reply was provided on Tuesday, 07 Mar 2006 12:39:45 Greenwich Mean Time.

   c. The document index.html was last modified on Saturday, 10 Dec 2005 18:27:46 GMT.

   d. There are 3874 bytes in the document being returned.

   e. The first five bytes of the returned document are: `<!doc`.

   f. The server agreed to a persistent connection, as indicated by the `Connection: Keep-Alive` field.

8. a. For starters, note that it takes 1 msec to send 100K bits over a 100 Mbps link. The delays associated with this scenario are:

   - 300 msec (RTT) to set up the TCP connection that will be used to request the base file.
   - 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client (for a total of 301 msec).
   - 300 msec (RTT) to set up TCP connection that will be used to request the img.01.jpg file.
   - 150 msec (one way delay) to send the GET message for img01.jpg and have it propagate to the server, plus 1 msec to transmit the img01.jpg

file, plus 150 msec for the img01.jpg file to propagate back to the client (for a total of 301 msec).

The last two steps above are repeated for the nine image files img02.jpg through img10.jpg. The total response time is therefore
$300 + 11 \times 601$, or 6.911 seconds.

b.
- 300 msec (RTT) to set up the TCP connection that will be used to request the base file.
- 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client (for a total of 301 msec).
- The client now sets up 10 parallel TCP connections. 300 msec (RTT) is needed to set up the 10 TCP connections (since they are set up in parallel).
- 150 msec (one way delay) to send the 10 GET messages in parallel for img01.jpg through img10.jpg and have the GET messages propagate to the server. It will take the server 10 msec to transmit the 10 jpeg files, plus 150 msec for the last jpeg file to propagate back to the client (for a total of 310 msec).

Putting this all together, the response time is

$$300 + 301 + 300 + 310 = 1.211 \text{ seconds.}$$

c.
- 300 msec (RTT) to set up the TCP connection that will be used to request the base file, and 10 images.
- 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client (for a total of 301 msec).
- 150 msec (one way delay) to send the GET message for img01.jpg and have it propagate to the server, plus 1 msec to transmit the img01.jpg file, plus 150 msec for the img01.jpg file to propagate back to the client (for a total of 301 msec).

The last step above is repeated for the nine image files img02.jpg through img10.jpg. The total response time is therefore

$$300 + 11 \times 301 = 3.611 \text{ seconds}$$

d.
- 300 msec (RTT) to set up the TCP connection that will be used to request the base file, and 10 images.
- 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client (for a total of 301 msec).

- 150 msec (one way delay) to send the 10 GET messages serially for img01.jpg through img10.jpg and have the GET messages propagate to the server (recall that we are assuming the GET message has zero transmission time). It will take the server 10 msec to transmit the 10 jpeg files, plus 150 msec for the last jpeg file to propagate back to the client (for a total of 310 msec).

The total response time is 300 + 301 + 310 = 911 msec.

9. a.
   - m1.a.com needs to resolve the name www.b.com to an IP address so it sends a DNS REQUEST message to its local DNS resolver.
   - The local DNS server does not have any information so it contacts a root DNS server with a REQUEST message.
   - The root DNS server returns name of DNS Top Level Domain server for .com.
   - The local DNS server contacts the .com TLD.
   - The TLD .com server returns the authoritative name server for b.com.
   - The local DNS server contacts the authoritative name server for b.com.
   - The authoritative name server for b.com returns the IP address of www.b1.com.
   - The HTTP client sends a HTTP GET message to www.b1.com, which is redirected by the client browser to the HTTP cache in the a.com network.
   - The HTTP cache does not find the requested document in its cache, so it sends the GET request to www.b.com.
   - www.b.com receives the GET request and sends the file from www.b.com to R2.
   - The 1 Gbit file is transmitted over the 1 Mbps link between R2 and R1 to the HTTP cache.
   - The 1 Gbit file is sent from the HTTP cache to m1.a.com.

   b. Let $t = 0$ be the time at which the user enters www.b.com into the browser

   - The HTTP client will send its HTTP GET message to www.b1.com through the local HTTP cache in a.com. This takes no time given the assumptions above.
   - The HTTP cache does not find the requested document in its cache. Therefore it must request the document from b.com. Before it can send a GET request to www.b.com, it must find out the IP address for www.b.com. To do this, it will have to use the DNS. To resolve the name www.b.com to an IP address, the Web cache sends a DNS REQUEST message to its local DNS resolver. This takes no time given the assumptions above.

- The local DNS server does not have any information so it contacts a root DNS server with a REQUEST message. This takes .5 sec given the assumptions above. At $t = 500$ msec, the root DNS server receives the message.
- The root DNS server returns the name of the DNS Top Level Domain server for .com. This takes 500 msec given the assumptions above. At $t = 1$ sec, the local DNS server receives the message and now knows the address of the .com TLD DNS server.
- The local DNS server contacts the .com TLD DNS server. This takes 500 msec given the assumptions above. At $t = 1.5$, the TLD DNS server receives the message.
- The TLD .com server returns the authoritative DNS server for b.com. This takes 500 msec given the assumptions above. At $t = 2$, the local DNS server receives the message.
- The local DNS server contacts the authoritative name server for b.com. This takes 100 msec given the assumptions above. At $t = 2.1$, the authoritative DNS server for b.com receives the message.
- The authoritative name server for b.com returns the IP address of www.b1.com. This takes 100 msec, given the assumptions above. At $t = 2.2$, the local DNS server for a.com receives the message, and returns this message to the HTTP Web cache.
- The Web cache is now ready to request the document from www.b.com. It takes 200 msec to set up the TCP connection between the HTTP cache and www.b.com.
- At $t = 2.4$ sec, the Web cache sends the GET request to www.b.com. It takes 100 msec for the GET request to propagate to www.b.com, given the assumptions above.
- At $t = 2.5$ sec, www.b.com receives the GET request and immediately begins sending the file in reply. If we assume that as soon as the first bit reaches R2, it is forwarded over the link between R2 and R1, then this transmission delay can be ignored, since the transmission of the file from www.b.com is pipelined with the transmission of the file between R2 and R1.
- The 1 Gbit file must be transmitted over the 1 Mbps link between R2 and R1. This takes 1,000 seconds. There is an additional 100 msec propagation delay. Thus, at $t = 1002.6$ secs, the last bit of the file is received at R1. If we assume that R1 forwards packets to the HTTP cache as it receives them from R2, then the transmission delay between R1 and the cache can be ignored since it is pipelined with the transmission from R2 to R1.

- There is a 1 sec delay to send the 1 Gbps file from R1 to the HTTP cache. If we assume that as soon as the first few bits of the file arrive at the router, they are forwarded to the cache, then this delay can be ignored.
- There is a 1 sec delay to send the 1 Gbps file from the HTTP cache to m1.a.com. If we assume that as soon as the first few bits of the file arrive at the cache, they are forwarded to the m1.a.com, then this delay can be ignored.

Thus, the total delay is approximately 1002.6 seconds.

c.
- The HTTP client at m2.a.com will send its HTTP GET message to www.b1.com through the local HTTP cache in a.com. This takes no time given the assumptions above.
- The HTTP cache finds the requested document in its cache, so it sends a GET request with an If-Modified-Since to www.b.com. This takes 100 msec given the assumptions above. Note that the cache does not need to contact the DNS, assuming it has cached the IP address associated with www.b.com.
- www.b.com receives the GET request. The document has not changed, so www.b.com sends a short HTTP REPLY message to the HTTP cache in a.com indicating that the cached copy is valid. (This takes 100 msec given the assumptions.)
- There is a 1 sec delay to send the 1 Gbps file from the HTTP cache to m2.a.com.

Thus, the total delay is: $.1 + .1 + 1 = 1.2$ sec.

d. Since it takes approximately 1000 sec to send the file from R2 to R1, the maximum rate at which requests to send the file from b.com to a.com is 1 request every 1000 seconds, or an arrival rate of .001 requests/sec.

10. a. 2 RTT to get each image: 4(2 RTT) = 8 RTT

b. 2 RTT for base, 2 RTT for remaining three images: 4 RTT

c. 2 RTT for base; 1 RTT for remaining three images: 3 RTT

11. When we say that control messages are “in-band,” it means that control message and data messages may be interleaved with each other on the same connection. A single message may contain both control information and data. When we say that control messages are “out-of-band,” it means that control and data messages are carried on separate connections. HTTP, DNS, and SMTP have in-band control, while FTP has out-of-band control messages.

12. Parallel download would only share the 100K bandwidth among the 10 connections (each getting just 10K bits/sec) thus, there is no significant advantage here. With persistent HTTP we avoid the SYN and SYNACK exchange, but that only requires 2 seconds (1 second to send the 100 bit SYN message

over the 100 bps link, and 1 second to receive the ACK). Given that each object takes 101 seconds to send and receive the ACK, the use of pipelining gives only a 2 percent gain.

13. This is true for UDP since the UDP packet (i.e., the Datagram Packet data structure returned from the ServerSocket.read( ) call) contains the IP address of the sender of the UDP packet. This is false for a TCP socket since the socket only returns the byte stream sent by the client, but no identifying information about the client.

14. A Web browser will cache objects locally in its browser cache. An institution might also have a Web cache, which each browser contacts to satisfy its Web requests. If the requested Web page is not in the Web cache, the Web cache will request and receive the object from the origin server, cache the object locally (in case some other browser requests the object), and return the object to the request browser. When the local DNS gets a translation request for a name that it is not in its cache, it will obtain the name/address translation pair and then cache this value so that future requests for translating the same name can be satisfied by the local DNS server, without going out to the larger DNS system.

15. The MAIL FROM: in SMTP is a message from the SMTP client that identifies the sender of the mail message to the SMTP server. The From: on the mail message itself is NOT an SMTP message, but rather is just a line in the body of the mail message.

16. FM radio, broadcast TV, and newspapers are all push systems. Information is sent out via these media regardless of whether or not anyone is tuning in to the TV or radio stations, or buying the newspaper. Also, content is not individualized for the receiver. The Web is a pull system because data is not transmitted to the browser unless it is explicitly requested by the browser. Some automated phone response systems are non-Internet pull systems. For example, I may call a number to get my horoscope, and enter my birth date in order to have my horoscope read to me.

17. a. The advantage of sending the QueryHit message directly over a TCP connection from Bob to Alice is that the QueryHit message is routed by the underlying Internet without passing through intermediate peers; thus, the delay in sending the message from Bob to Alice should be substantially less. The disadvantage is that each peer that has a match would ask Alice to open a TCP connection; Alice may therefore have to opens tens or hundreds of TCP connections for a given query.

    b. When a QueryHit message enters a peer, the peer records the MessageID in a table along with an identifier of the TCP socket from which the message arrived. When the same peer receives a QueryHit message with the same MessageID, it indexes the table and determines the socket to which it should forward the message.

c. As the query message propagates through peers, each peer includes its IP address in the query message, creating a list of IP addresses. When there is a query match, the peer with the match includes the list in the QueryHit message. When a peer receives a QueryHit, it finds the preceding node in the list and forwards the message to that preceding peer.

18. The welcoming socket is used in a server to wait for incoming client connection requests in connection-oriented (TCP-based) communication. A new socket is created on return from the accept(). This new socket is then used by the server to communicate back to the client.

19. If all traffic is UDP, then the number of sockets will equal the number of ports. If there is TCP traffic, then the number of sockets will be greater, because each welcoming socket, and the socket created as a result of accept()ing a connection will have the same destination port number.

20. See page 155 of your textbook.