

Homework Problems and Questions

Chapter 2 Review Questions

SECTION 2.1

- R1. List five nonproprietary Internet applications and the application-layer protocols that they use.
- R2. What is the difference between network architecture and application architecture?
- R3. For a communication session between a pair of processes, which process is the client and which is the server?
- R4. For a P2P file-sharing application, do you agree with the statement, “There is no notion of client and server sides of a communication session”? Why or why not?
- R5. What information is used by a process running on one host to identify a process running on another host?
- R6. Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?
- R7. Referring to [Figure 2.4](#), we see that none of the applications listed in [Figure 2.4](#) requires both no data loss and timing. Can you conceive of an application that requires no data loss and that is also highly time-sensitive?
- R8. List the four broad classes of services that a transport protocol can provide. For each of the service classes, indicate if either UDP or TCP (or both) provides such a service.
- R9. Recall that TCP can be enhanced with SSL to provide process-to-process security services, including encryption. Does SSL operate at the transport layer or the application layer? If the application developer wants TCP to be enhanced with SSL, what does the developer have to do?

SECTION 2.2–2.5

- R10. What is meant by a handshaking protocol?
- R11. Why do HTTP, SMTP, and POP3 run on top of TCP rather than on UDP?
- R12. Consider an e-commerce site that wants to keep a purchase record for each of its customers. Describe how this can be done with cookies.
- R13. Describe how Web caching can reduce the delay in receiving a requested object. Will Web caching reduce the delay for all objects requested by a user or for only some of the objects?

Why?

R14. Telnet into a Web server and send a multiline request message. Include in the request message the *If-modified-since:* header line to force a response message with the *304 Not Modified* status code.

R15. List several popular messaging apps. Do they use the same protocols as SMS?

R16. Suppose Alice, with a Web-based e-mail account (such as Hotmail or Gmail), sends a message to Bob, who accesses his mail from his mail server using POP3. Discuss how the message gets from Alice's host to Bob's host. Be sure to list the series of application-layer protocols that are used to move the message between the two hosts.

R17. Print out the header of an e-mail message you have recently received. How many *Received:* header lines are there? Analyze each of the header lines in the message.

R18. From a user's perspective, what is the difference between the download-and-delete mode and the download-and-keep mode in POP3?

R19. Is it possible for an organization's Web server and mail server to have exactly the same alias for a hostname (for example, *foo.com*)? What would be the type for the RR that contains the hostname of the mail server?

R20. Look over your received e-mails, and examine the header of a message sent from a user with a .edu e-mail address. Is it possible to determine from the header the IP address of the host from which the message was sent? Do the same for a message sent from a Gmail account.

SECTION 2.5

R21. In BitTorrent, suppose Alice provides chunks to Bob throughout a 30-second interval. Will Bob necessarily return the favor and provide chunks to Alice in this same interval? Why or why not?

R22. Consider a new peer Alice that joins BitTorrent without possessing any chunks. Without any chunks, she cannot become a top-four uploader for any of the other peers, since she has nothing to upload. How then will Alice get her first chunk?

R23. What is an overlay network? Does it include routers? What are the edges in the overlay network?

SECTION 2.6

R24. CDNs typically adopt one of two different server placement philosophies. Name and briefly describe them.

R25. Besides network-related considerations such as delay, loss, and bandwidth performance, there are other important factors that go into designing a CDN server selection strategy. What are they?

SECTION 2.7

R26. In Section 2.7, the UDP server described needed only one socket, whereas the TCP server needed two sockets. Why? If the TCP server were to support n simultaneous connections, each from a different client host, how many sockets would the TCP server need?

R27. For the client-server application over TCP described in [Section 2.7](#), why must the server program be executed before the client program? For the client-server application over UDP, why may the client program be executed before the server program?

Problems

P1. True or false?

- A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.
- Two distinct Web pages (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.
- With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.
- The `Date:` header in the HTTP response message indicates when the object in the response was last modified.
- HTTP response messages never have an empty message body.

P2. SMS, iMessage, and WhatsApp are all smartphone real-time messaging systems. After doing some research on the Internet, for each of these systems write one paragraph about the protocols they use. Then write a paragraph explaining how they differ.

P3. Consider an HTTP client that wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What transport and application-layer protocols besides HTTP are needed in this scenario?

P4. Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., 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 answer.

```
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<cr><lf>Accept-
Encoding: zip, deflate<cr><lf>Accept-Charset: ISO
-8859-1, utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

- a. What is the URL of the document requested by the browser?
- b. What version of HTTP is the browser running?
- c. Does the browser request a non-persistent or a persistent connection?
- d. What is the IP address of the host on which the browser is running?
- e. What type of browser initiates this message? Why is the browser type needed in an HTTP request message?

P5. The text below shows the reply sent from the server in response to the HTTP GET message in the question above. Answer the following questions, indicating where in the message below you find the answer.

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 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-ST550ASpring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>
```

- a. Was the server able to successfully find the document or not? What time was the document reply provided?
- b. When was the document last modified?
- c. How many bytes are there in the document being returned?
- d. What are the first 5 bytes of the document being returned? Did the server agree to a

persistent connection?

P6. Obtain the HTTP/1.1 specification (RFC 2616). Answer the following questions:

- a. Explain the mechanism used for signaling between the client and server to indicate that a persistent connection is being closed. Can the client, the server, or both signal the close of a connection?
- b. What encryption services are provided by HTTP?
- c. Can a client open three or more simultaneous connections with a given server?
- d. Either a server or a client may close a transport connection between them if either one detects the connection has been idle for some time. Is it possible that one side starts closing a connection while the other side is transmitting data via this connection? Explain.

P7. Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT_1, \dots, RTT_n . Further suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Let RTT_0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object?

P8. Referring to Problem P7, suppose the HTML file references eight very small objects on the same server. Neglecting transmission times, how much time elapses with

- a. Non-persistent HTTP with no parallel TCP connections?
- b. Non-persistent HTTP with the browser configured for 5 parallel connections?
- c. Persistent HTTP?

P9. Consider [Figure 2.12](#), for which there is an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institution's browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is three seconds on average (see [Section 2.2.5](#)). Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use $\Delta/(1-\Delta\beta)$, where Δ is the average time required to send an object over the access link and β is the arrival rate of objects to the access link.

- a. Find the total average response time.
- b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.

P10. Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 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 100 Kbits 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? Justify and explain your answer.

P11. Consider the scenario introduced in the previous problem. Now suppose that the link is shared by Bob with four other users. Bob uses parallel instances of non-persistent HTTP, and the other four users use non-persistent HTTP without parallel downloads.

- a. Do Bob's parallel connections help him get Web pages more quickly? Why or why not?
- b. If all five users open five parallel instances of non-persistent HTTP, then would Bob's parallel connections still be beneficial? Why or why not?

P12. Write a simple TCP program for a server that accepts lines of input from a client and prints the lines onto the server's standard output. (You can do this by modifying the `TCPServer.py` program in the text.) Compile and execute your program. On any other machine that contains a Web browser, set the proxy server in the browser to the host that is running your server program; also configure the port number appropriately. Your browser should now send its GET request messages to your server, and your server should display the messages on its standard output. Use this platform to determine whether your browser generates conditional GET messages for objects that are locally cached.

P13. What is the difference between `MAIL FROM:` in SMTP and `From:` in the mail message itself?

P14. How does SMTP mark the end of a message body? How about HTTP? Can HTTP use the same method as SMTP to mark the end of a message body? Explain.

P15. Read RFC 5321 for SMTP. What does MTA stand for? Consider the following received spam e-mail (modified from a real spam e-mail). Assuming only the originator of this spam e-mail is malicious and all other hosts are honest, identify the malicious host that has generated this spam e-mail.

```
From - Fri Nov 07 13:41:30 2008
Return-Path: <tennis5@pp33head.com>
Received: from barmail.cs.umass.edu (barmail.cs.umass.edu
[128.119.240.3]) by cs.umass.edu (8.13.1/8.12.6) for
<hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:10 -0500
Received: from asusus-4b96 (localhost [127.0.0.1]) by
barmail.cs.umass.edu (Spam Firewall) for <hg@cs.umass.edu>; Fri, 7
```

```
Nov 2008 13:27:07 -0500 (EST)
Received: from asusus-4b96 ([58.88.21.177]) by barmail.cs.umass.edu
for <hg@cs.umass.edu>; Fri, 07 Nov 2008 13:27:07 -0500 (EST)
Received: from [58.88.21.177] by inbnd55.exchangeddd.com; Sat, 8
Nov 2008 01:27:07 +0700
From: "Jonny" <tennis5@pp33head.com>
To: <hg@cs.umass.edu>

Subject: How to secure your savings
```

P16. Read the POP3 RFC, RFC 1939. What is the purpose of the UIDL POP3 command?

P17. Consider accessing your e-mail with POP3.

- a. Suppose you have configured your POP mail client to operate in the download-and-delete mode. Complete the following transaction:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: .....blah
S: .
?
?
```

- b. Suppose you have configured your POP mail client to operate in the download-and-keep mode. Complete the following transaction:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: .....blah
S: .
?
```


- c. Suppose you have configured your POP mail client to operate in the download-and-keep mode. Using your transcript in part (b), suppose you retrieve messages 1 and 2, exit POP, and then five minutes later you again access POP to retrieve new e-mail. Suppose that in the five-minute interval no new messages have been sent to you. Provide a transcript of this second POP session.

P18.

- a. What is a *whois* database?
- b. Use various whois databases on the Internet to obtain the names of two DNS servers. Indicate which whois databases you used.
- c. Use nslookup on your local host to send DNS queries to three DNS servers: your local DNS server and the two DNS servers you found in part (b). Try querying for Type A, NS, and MX reports. Summarize your findings.
- d. Use nslookup to find a Web server that has multiple IP addresses. Does the Web server of your institution (school or company) have multiple IP addresses?
- e. Use the ARIN whois database to determine the IP address range used by your university.
- f. Describe how an attacker can use whois databases and the nslookup tool to perform reconnaissance on an institution before launching an attack.
- g. Discuss why whois databases should be publicly available.

P19. In this problem, we use the useful *dig* tool available on Unix and Linux hosts to explore the hierarchy of DNS servers. Recall that in **Figure 2.19**, a DNS server in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server. First read the man page for *dig*, and then answer the following questions.

- a. Starting with a root DNS server (from one of the root servers [a-m].root-servers.net), initiate a sequence of queries for the IP address for your department's Web server by using *dig*. Show the list of the names of DNS servers in the delegation chain in answering your query.
- b. Repeat part (a) for several popular Web sites, such as google.com, yahoo.com, or amazon.com.

P20. Suppose you can access the caches in the local DNS servers of your department. Can you propose a way to roughly determine the Web servers (outside your department) that are most popular among the users in your department? Explain.

P21. Suppose that your department has a local DNS server for all computers in the department.

You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

P22. Consider distributing a file of $F=15$ Gbits to N peers. The server has an upload rate of $u_s=30$ Mbps, and each peer has a download rate of $d_i=2$ Mbps and an upload rate of u . For $N=10, 100$, and $1,000$ and $u=300$ Kbps, 700 Kbps, and 2 Mbps, prepare a chart giving the minimum distribution time for each of the combinations of N and u for both client-server distribution and P2P distribution.

P23. Consider distributing a file of F bits to N peers using a client-server architecture. Assume a fluid model where the server can simultaneously transmit to multiple peers, transmitting to each peer at different rates, as long as the combined rate does not exceed u_s .

- Suppose that $u_s/N \leq d_{\min}$. Specify a distribution scheme that has a distribution time of NF/u_s .
- Suppose that $u_s/N \geq d_{\min}$. Specify a distribution scheme that has a distribution time of F/d_{\min} .
- Conclude that the minimum distribution time is in general given by $\max\{NF/u_s, F/d_{\min}\}$.

P24. Consider distributing a file of F bits to N peers using a P2P architecture. Assume a fluid model. For simplicity assume that d_{\min} is very large, so that peer download bandwidth is never a bottleneck.

- Suppose that $u_s \leq (u_s + u_1 + \dots + u_N)/N$. Specify a distribution scheme that has a distribution time of F/u_s .
- Suppose that $u_s \geq (u_s + u_1 + \dots + u_N)/N$. Specify a distribution scheme that has a distribution time of $NF/(u_s + u_1 + \dots + u_N)$.
- Conclude that the minimum distribution time is in general given by $\max\{F/u_s, NF/(u_s + u_1 + \dots + u_N)\}$.

P25. Consider an overlay network with N active peers, with each pair of peers having an active TCP connection. Additionally, suppose that the TCP connections pass through a total of M routers. How many nodes and edges are there in the corresponding overlay network?

P26. Suppose Bob joins a BitTorrent torrent, but he does not want to upload any data to any other peers (so called free-riding).

- Bob claims that he can receive a complete copy of the file that is shared by the swarm. Is Bob's claim possible? Why or why not?
- Bob further claims that he can further make his "free-riding" more efficient by using a collection of multiple computers (with distinct IP addresses) in the computer lab in his department. How can he do that?

P27. Consider a DASH system for which there are N video versions (at N different rates and qualities) and N audio versions (at N different rates and qualities). Suppose we want to allow the

player to choose at any time any of the N video versions and any of the N audio versions.

- a. If we create files so that the audio is mixed in with the video, so server sends only one media stream at given time, how many files will the server need to store (each a different URL)?
- b. If the server instead sends the audio and video streams separately and has the client synchronize the streams, how many files will the server need to store?

P28. Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

- a. Suppose you run TCPClient before you run TCPServer. What happens? Why?
- b. Suppose you run UDPClient before you run UDPServer. What happens? Why?
- c. What happens if you use different port numbers for the client and server sides?

P29. Suppose that in UDPClient.py, after we create the socket, we add the line:

```
clientSocket.bind('', 5432))
```

Will it become necessary to change UDPServer.py? What are the port numbers for the sockets in UDPClient and UDPServer? What were they before making this change?

P30. Can you configure your browser to open multiple simultaneous connections to a Web site? What are the advantages and disadvantages of having a large number of simultaneous TCP connections?

P31. We have seen that Internet TCP sockets treat the data being sent as a byte stream but UDP sockets recognize message boundaries. What are one advantage and one disadvantage of byte-oriented API versus having the API explicitly recognize and preserve application-defined message boundaries?

P32. What is the Apache Web server? How much does it cost? What functionality does it currently have? You may want to look at Wikipedia to answer this question.

Socket Programming Assignments

The Companion Website includes six socket programming assignments. The first four assignments are summarized below. The fifth assignment makes use of the ICMP protocol and is summarized at the end of **Chapter 5**. The sixth assignment employs multimedia protocols and is summarized at the end of **Chapter 9**. It is highly recommended that students complete several, if not all, of these assignments. Students can find full details of these assignments, as well as important snippets of the Python code, at the Web site www.pearsonhighered.com/cs-resources.

Assignment 1: Web Server

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will (i) create a connection socket when contacted by a client (browser); (ii) receive the HTTP request from this connection; (iii) parse the request to determine the specific file being requested; (iv) get the requested file from the server's file system; (v) create an HTTP response message consisting of the requested file preceded by header lines; and (vi) send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

In the Companion Website, we provide the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web server.

Assignment 2: UDP Pinger

In this programming assignment, you will write a client ping program in Python. Your client will send a simple ping message to a server, receive a corresponding pong message back from the server, and determine the delay between when the client sent the ping message and received the pong message. This delay is called the Round Trip Time (RTT). The functionality provided by the client and server is similar to the functionality provided by standard ping program available in modern operating systems. However, standard ping programs use the Internet Control Message Protocol (ICMP) (which we will study in [Chapter 5](#)). Here we will create a nonstandard (but simple!) UDP-based ping program.

Your ping program is to send 10 ping messages to the target server over UDP. For each message, your client is to determine and print the RTT when the corresponding pong message is returned. Because UDP is an unreliable protocol, a packet sent by the client or server may be lost. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and print a message accordingly.

In this assignment, you will be given the complete code for the server (available in the Companion Website). Your job is to write the client code, which will be very similar to the server code. It is recommended that you first study carefully the server code. You can then write your client code, liberally cutting and pasting lines from the server code.

Assignment 3: Mail Client

The goal of this programming assignment is to create a simple mail client that sends e-mail to any recipient. Your client will need to establish a TCP connection with a mail server (e.g., a Google mail server), dialogue with the mail server using the SMTP protocol, send an e-mail message to a recipient

(e.g., your friend) via the mail server, and finally close the TCP connection with the mail server.

For this assignment, the Companion Website provides the skeleton code for your client. Your job is to complete the code and test your client by sending e-mail to different user accounts. You may also try sending through different servers (for example, through a Google mail server and through your university mail server).

Assignment 4: Multi-Threaded Web Proxy

In this assignment, you will develop a Web proxy. When your proxy receives an HTTP request for an object from a browser, it generates a new HTTP request for the same object and sends it to the origin server. When the proxy receives the corresponding HTTP response with the object from the origin server, it creates a new HTTP response, including the object, and sends it to the client. This proxy will be multi-threaded, so that it will be able to handle multiple requests at the same time.

For this assignment, the Companion Website provides the skeleton code for the proxy server. Your job is to complete the code, and then test it by having different browsers request Web objects via your proxy.

Wireshark Lab: HTTP

Having gotten our feet wet with the Wireshark packet sniffer in Lab 1, we're now ready to use Wireshark to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic GET/reply interaction, HTTP message formats, retrieving large HTML files, retrieving HTML files with embedded URLs, persistent and non-persistent connections, and HTTP authentication and security.

As is the case with all Wireshark labs, the full description of this lab is available at this book's Web site, www.pearsonhighered.com/cs-resources.

Wireshark Lab: DNS

In this lab, we take a closer look at the client side of the DNS, the protocol that translates Internet hostnames to IP addresses. Recall from [Section 2.5](#) that the client's role in the DNS is relatively simple—a client sends a query to its local DNS server and receives a response back. Much can go on under the covers, invisible to the DNS clients, as the hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client's DNS query. From the DNS client's standpoint, however, the protocol is quite simple—a query is formulated to the local DNS server and a response is received from that server. We observe DNS in action in this lab.