Equipped with knowledge about Internet application structure and application-level protocols, we're now ready to head further down the protocol stack and examine the transport layer in Chapter 3.

Homework Problems and Questions

Chapter 2 Review Questions

SECTION 2.1

R.1. List five nonproprietary Internet applications and the application-layer protocols that they use.

R.2. What is the difference between network architecture and application architecture?

R.3. For a communication session between a pair of processes, which process is the client and which is the server?

R.4. 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?

R.5. What information is used by a process running on one host to identify a process running on another host?

R.6. 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?

R.7. 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?

R.8. 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.

R.9. 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?

SECTIONS 2.2–2.5

R.10. What is meant by a handshaking protocol?

R.11. Why do HTTP, FTP, SMTP, and POP3 run on top of TCP rather than on UDP?

R.12. 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.

R.13. 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?

R.14. 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.

R.15. Why is it said that FTP sends control information "out-of-band"?

R.16. 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.

R.17. 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.

R.18. From a user's perspective, what is the difference between the download-and-delete mode and the download-and-keep mode in POP3?

R.19. 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?

SECTION 2.6

R.20. 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?

R.21. 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?

R.22. What is an overlay network? Does it include routers? What are the edges in the overlay network? How is the query-flooding overlay network created and maintained?

R.23. In what way is instant messaging with a centralized index a hybrid of client-server and P2P architectures?

R.24. Consider a DHT with a mesh overlay topology (that is, every peer tracks all peers in the system). What are the advantages and disadvantages of such a design? What are the advantages and disadvantages of a circular DHT (with no shortcuts)?

R.25. Skype uses P2P techniques for two important functions. What are they?

R.26. List at least four different applications that are naturally suitable for P2P architectures. (*Hint:* File distribution and instant messaging are two.)

SECTIONS 2.7–2.8

R.27. The UDP server described in Section 2.8 needed only one socket, whereas the TCP server described in Section 2.7 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?

R.28. 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 described in Section 2.8, why may the client program be executed before the server program?

Problems

P1. True or false?

a. 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.

b. 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.

c. With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.

d. The `Date:` header in the HTTP response message indicates when the object in the response was last modified.

e. HTTP response messages never have an empty message body.

P2. Read RFC 959 for FTP. List all of the client commands that are supported by the RFC.

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 italized 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-ST550A Spring 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 $\text{RTT}_1, \ldots, \text{RTT}_n$. 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.java 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 email (modified from a real spam email). Assuming only the originator of this spam email is malacious and all other hosts are honest, identify the malacious host that has generated this spam email.

```
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 tool *dig* tool available on Unix and Linux hosts to explore the hierarchy of DNS servers. Recall that in Figure 2.21, a DNS server higher 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 come up a way to determine if an external Web site was very 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.

a. Suppose that $u_s/N \le d_{\min}$. Specify a distribution scheme that has a distribution time of NF/u_s.

b. Suppose that $u_s/N \ge d_{\min}$. Specify a distribution scheme that has a distribution time of $F/ d_{\min}$.

c. 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.

a. Suppose that $u_s \le (u_s + u_1 + \ldots + u_N)/N$. Specify a distribution scheme that has a distribution time of F/u_s.

b. Suppose that $u_s \ge (u_s + u_1 + \ldots + u_N)/N$. Specify a distribution scheme that has a distribution time of $NF/(u_s + u_1 + \ldots + u_N)$.

c. Conclude that the minimum distribution time is in general given by $\max\{F/u_s, NF/(u_s + u_1 + \ldots + u_N)\}$.

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).

a. 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?

b. 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. In this problem, we are interested in finding out the efficiency of a BitTorrent-like P2P file sharing system. Consider two peers Bob and Alice. They join a torrent with M peers in total (including Bob and Alice) that are sharing a file consisting of N chunks. Assume that at a particular time t, the chunks that a peer has are uniformly at random chosen from all N chunks, and no peer has all N chunks. Answer the following questions.

a. What is the probability that Bob has all the chunks that Alice has, given that the numbers of chunks that Bob and Alice have are denoted by n_b and n_a?

b. Remove part of the conditioning in part a) to find out the probability that Bob has all the chunks that Alice has, given that Alice has n_a chunks?

c. Suppose that each peer in BitTorrent has 5 neighbors. What is the probability that Bob has data that is of interest to at least one of his five neighbors?

P28. In the circular DHT example in Section 2.6.2, suppose that peer 3 learns that peer 5 has left. How does peer 3 update its successor state information? Which peer is now its first successor? Its second successor?

P29. In the circular DHT example in Section 2.6.2, suppose that a new peer 6 wants to join the DHT and peer 6 initially only knows peer 15's IP address. What steps are taken?

P30. Consider a circular DHT with node and key identifiers in the range [0, 63]. Suppose there are eight peers with identifiers 0, 8, 16, 24, 32, 40, 48, and 56.

a. Suppose each peer can have one shortcut peer. For each of the eight peers, determine its shortcut peer so that the number of messages sent for any query (beginning at any peer) is minimized.

b. Repeat (a) but now allow each peer to have two shortcut peers.

P31. Because an integer in $[0, 2^n - 1]$ can be expressed as an n-bit binary number in a DHT, each key can be expressed as $k = (k_0, k_1, \ldots, k_{n-1})$, and each peer identifier can be expressed $p = (p_0, p_1, \ldots, p_{n-1})$. Let's now define the XOR distance between a key k and peer p as

$$d(k, p) = \sum_{j=0}^{n-1} | k_j - p_j | 2^j$$

Describe how this metric can be used to assign (key, value) pairs to peers. (To learn about how to build an efficient DHT using this natural metric, see [Maymounkov 2002] in which the Kademlia DHT is described.)

P32. Consider a generalized version of the scheme described in the previous problem. Instead of using binary numbers, we now treat key and peer identifiers as base-b numbers where $b > 2$, and then use the metric in the previous problem to design a DHT (with 2 replace with b). Compare this DHT based on base-b numbers with the DHT based on binary numbers. In the worst case, which DHT generates more messages per query? Why?

P33. As DHTs are overlay networks, they may not necessarily match the underlay physical network well in the sense that two neighboring peers might be physically very far away; for example, one peer could be in Asia and its neighbor could be in North America. If we randomly and uniformly assign identifiers to newly joined peers, would this assignment scheme cause such a mismatch? Explain. And how would such a mismatch affect the DHT's performance?

P34. Install and compile the Java 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?

P35. Suppose that in UDPClient.java we replace the line

```
DatagramSocket clientSocket = new DatagramSocket();
```

with

```
DatagramSocket clientSocket = new DatagramSocket(5432);
```

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

Discussion Questions

D1. Why do you think P2P file-sharing applications are so popular? Is it because they (debatably illegally) distribute free music and video? Is it because their massive number of servers efficiently responds to a massive demand for megabytes? Or is it all of these?

D2. Read the paper "The Darknet and the Future of Content Distribution" by Biddle, England, Peinado, and Willman [Biddle 2003]. Do you agree with all of the views of the authors? Why or why not?

D3. E-commerce sites and other Web sites often have back-end databases. How do HTTP servers communicate with these back-end databases?

D4. How can you configure your browser for local caching? What caching options do you have?

D5. 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?

D6. 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?

D7. What is the Apache Web server? How much does it cost? What functionality does it currently have?

D8. Many BitTorrent clients use DHTs to create a distributed tracker. For these DHTs, what is the "key" and what is the "value"?

D9. Suppose that the Web standards organizations decide to change the naming convention so that each object is named and referenced by a unique name that is location-independent (a so-called URN). Discuss some of the issues surrounding such a change.

D10. Are any companies distributing live television feeds over the Internet today? If so, are these companies using client-server or P2P architectures?

D11. Are companies today providing a video-on-demand service over the Internet using a P2P architecture?

D12. How does Skype provide a PC-to-phone service to many different destination countries?

D13. What are some of the most popular BitTorrent clients today?

Socket Programming Assignments

Assignment 1: Multi-Threaded Web Server

By the end of this programming assignment, you will have developed, in Java, a multithreaded Web server that is capable of serving multiple requests in parallel. You are going to implement version 1.0 of HTTP, as defined in RFC 1945.

HTTP/1.0 creates a separate TCP connection for each request/response pair. A separate thread handles each of these connections. There will also be a main thread, in which the server listens for clients that want to establish connections. To simplify the programming task, we will develop the code in two stages. In the first stage, you will write a multithreaded server that simply displays the contents of the HTTP request message that it receives. After this program is running properly, you will add the code required to generate an appropriate response.

As you develop the code, you can test your server with a Web browser. But remember that you are not serving through the standard port 80, so you need to specify the port number within the URL that you give your browser. For example, if your host's name is `host.someschool.edu`, your server is listening to port 6789, and you want to retrieve the file `index.html`, then you would specify the following URL within the browser:

```
http://host.someschool.edu:6789/index.html
```

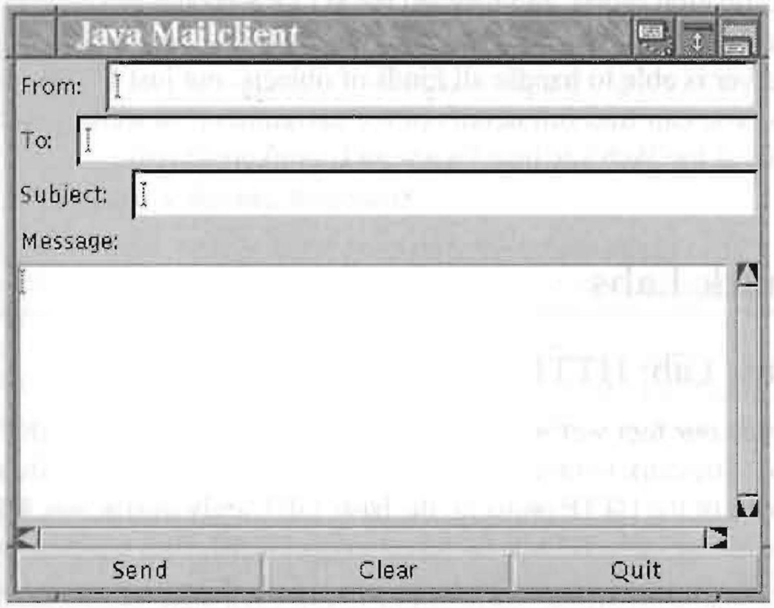
When your server encounters an error, it should send a response message with an appropriate HTML source so that the error information is displayed in the browser window. You can find full details of this assignment, as well as important snippets of Java code, at the Web site http://www.awl.com/kurose-ross.

Assignment 2: Mail Client

In this assignment, you will develop in Java a mail user agent with the following characteristics:

- Provides a graphical interface for the sender, with fields for the local mail server, sender's e-mail address, recipient's e-mail address, subject of the message, and the message itself.
- Establishes a TCP connection between the mail client and the local mail server. Sends SMTP commands to local mail server. Receives and processes SMTP commands from local mail server.

Here is what your interface will look like:



You will develop the user agent so it sends an e-mail message to at most one recipient at a time. Furthermore, the user agent will assume that the domain part of the recipient's e-mail address is the canonical name of the recipient's SMTP server. (The user agent will not perform a DNS lookup for an MX record, so the sender must supply the actual name of the mail server.) You can find full details of the assignment, as well important snippets of Java code, at the Web site http://www.awl.com/kurose-ross.

Assignment 3: UDP Pinger Lab

In this lab, you will implement a simple UDP-based Ping client and server. The functionality provided by these programs is similar to the standard Ping program available in modern operating systems. Standard Ping works by sending Internet Control Message Protocol (ICMP) ECHO messages, which the remote machine echoes back to the sender. The sender can then determine the round-trip time between itself and the computer it pinged.

Java does not provide any functionality to send or receive ICMP messages, which is why in this lab you will implement the pinging in the application layer with standard UDP sockets and messages. You can find full details of the assignment, as well important snippets of Java code, at the Web site http://www.awl.com/kurose-ross.

Assignment 4: Web Proxy Server

In this lab you'll develop a simple Web proxy server, which is also able to cache Web pages. This server will accept a GET message from a browser, forward the GET message to the destination Web server, receive the HTTP response message from the destination server, and forward the HTTP response message to the browser. This is a very simple proxy server; it only understands simple GET requests. However, the server is able to handle all kinds of objects, not just HTML pages, including images. You can find full details of the assignment, as well important snippets of Java code, at the Web site http://www.awl.com/kurose-ross.

Wireshark Labs

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, http://www.awl.com/kurose-ross.

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.

The full description of this lab is available at this book's Web site, http://www.awl.com/kurose-ross.