CHAPTER 1

WHAT IS A WEB SERVER?

Never trust a computer you can't throw out a window.

—Steve Wozniak

CHAPTER OBJECTIVES

In this chapter you will learn about:

In this chapter we provide some background information on how documents are published on the World Wide Web. We explain how computers on the Internet talk to each other and, more important, how Web pages get from a Web server to a browser. When setting up a Web server, it is important to know a little about the underlying technology: the communications protocols, network terminology, and document formats.

LAB 1.1

LAB 1.1

CLIENT/SERVER BASICS

LAB OBJECTIVES

After completing this lab, you will be able to:

- Understand Client/Server Concepts
- Describe Basic Functionality of Web Servers and Browsers

Before we even start to talk about Web servers, let's look at clients and servers in general. In network terminology, a client is a piece of hardware or software used to communicate with a data provider (server). Normally, only one user uses a specific client at a time. A client connects to a server to send and receive information. Think of a client as a program that gets information from somewhere else. A server is usually a large computer capable of providing data to many clients at the same time. The word *server* can mean the physical computer or piece of hardware, or it can refer to the actual server software or daemon running on that machine. A *daemon* is a program that offers a service to other programs, usually over a network. It accepts requests from clients, processes the requests, and returns the results to the requesting client. Although the client and server can be on the same machine, they are usually on separate machines connected by some kind of network.

The World Wide Web (WWW) uses this client/server model to allow millions of users to access Web sites all over the world. A Web server is a specific type of server that knows how to communicate with clients using the HyperText Transfer Protocol (HTTP). A protocol is just a standard set of rules that allow a client and server to communicate. For a client and server to communicate, they must speak the same protocol. HTTP allows clients to request documents and servers to respond with those documents. We will look at HTTP in more detail in Lab 1.3, but for now, think

of it as a small language. On the Web, the clients are Web browsers—applications especially well suited for displaying HTML content. Web servers wait for clients to connect and when a connection is established, they receive a request from the client and then respond—usually returning a document or image. The Web server process is usually referred to as the HTTPD, or HTTP daemon.

NETWORK CONNECTIONS AND PORTS

To connect to a server, the client must be able to communicate with it over the network. Computers connected to the Internet typically communicate using TCP/IP (Transmission Control Protocol and the Internet Protocol). TCP/IP allows different types of computers to communicate at a low level; it is up to applications, however, to determine how client and server software talk to each other. Applications such as e-mail, ftp, and Web browsers use their own protocols (SMTP, HTTP, etc.) to communicate on the application level while using TCP/IP at the network level.

TCP/IP uses IP addresses to communicate between computers. Each computer on the Internet has its own unique IP address. When a computer wants to send a message to another machine on the Internet, it specifies the address of the other machine and the message finds its way through the network. This is similar to how a letter finds its way through the postal system. The destination computer may have many different services running on it, so to specify which service we want to communicate with, we must use a port number. Each service has a unique number assigned to it known as a *port number*. Most of the services have standard port numbers.

SERVERS AND BROWSERS

The main goal of any Web server is to provide documents to clients. The first Web servers were very simple and did little more than this. Today's Web servers are full of features that allow them to do more than just respond to simple requests for static documents, and many provide easy-to-use graphical user interfaces for administration and customization. Today's servers support options that allow the creation of dynamic documents—documents that are generated on the fly, not stored on disk.

The purpose of a Web browser is to retrieve and display information from a Web server by using HTTP. A browser allows any user to access a server easily. Without even knowing what a Web server is, a user can easily obtain information from one just by entering a URL. Browsers have evolved

also, adding features that far extend the capabilities of browsers that once displayed only basic HTML.

BROWSER PLUG-INS

A plug-in extends the capabilities of a browser by allowing it to display more than just HTML documents. Adobe's Acrobat plug-in allows browsers to display PDF (Portable Document Format) files, and Macromedia Shockwave and Flash plug-ins allow authors to embed multimedia applications in Web pages. Plug-ins typically rely on the browser to retrieve the content (using HTTP) and then display it themselves. Plug-ins such as Real Networks' RealPlayer, however, are able to use their own protocols instead of HTTP to retrieve content. RealPlayer enables browsers to play streaming audio and video, which has different requirements than text documents, so a protocol other than HTTP is used by the plug-in to enhance performance. Helper applications are similar to plug-ins; they allow you to view content that your browser cannot. Unlike plug-ins, helper applications run outside the browser. They are stand-alone applications and they cannot be used to embed content in Web pages.

LAB 1.1 EXERCISES

1.1.1 UNDERSTAND CLIENT/SERVER CONCEPTS

a) What are the benefits of using a client/server model?

b) Give an example of another type of client/server application.

c) How does a hostname get translated into an address? Find out the IP address of a host (try `www.phptr.com`).

1.1.2 DESCRIBE BASIC FUNCTIONALITY OF WEB SERVERS AND BROWSERS

a) What is the primary function of an HTTP server?

b) Who developed the first Web server? What other early Web servers were developed?

c) What is the primary function of a Web browser?

d) What was the first Web browser? Why did it succeed where similar services (such as ftp, gopher, and WAIS) failed?

LAB 1.1 EXERCISE ANSWERS

1.1.1 UNDERSTAND CLIENT/SERVER CONCEPTS

a) What are the benefits of using a client/server model?

Answer: Making data available on a server can make it possible for many clients to access that data. Clients can be dispersed geographically. Clients are sure to receive the most up-to-date information. The framework of the server can be changed (database back ends can be switched) without affecting the clients. Server maintenance is easier if all clients are connecting to one place.

The client/server model is ideal for distributed applications. A server allows clients access to current data and allows clients to be dispersed anywhere there is network connectivity. A client generally asks a server for a

resource but does not care how the server gets that resource. Therefore, the server's underlying technology can be changed without changing the client's functionality. For instance, you could change your server to access an Oracle database instead of a Microsoft database. Another benefit to having all services provided through a central server is that maintaining those services becomes a little easier—or at least more manageable.

One of the benefits of this model is that all account information is located in a central place. Consider the example of a bank with automated teller machines (ATMs). If bank account information were stored at each ATM site, it would be much harder to keep accounts up to date. By centralizing account information, many clients are able to get up-to-date account information easily. Administration is also easier when there is just one central server to worry about. It is easier to monitor and maintain one server or even a number of servers when they are all in one centralized location.

b) Give an example of another type of client/server application.

Answer: A classic example of client/server is a bank ATM network. Think of the ATMs as clients—one user at a time can use each ATM to make withdrawals from their account. Each ATM connects to a central computer (a server) to verify your PIN number and gain access to your account information.

c) How does a hostname get translated into an address? Find out the IP address of a host (try `www.phptr.com`).

Answer: When a client wants to talk to a server, it must know the IP address. A user will usually enter a hostname rather than IP address, though, and the computer will then resolve the hostname into an IP address that it can use. When the client makes an initial request to talk with a server, it specifies which IP address it wants to talk with (the unique IP address of the server) and specifies a port number. A port number is used to specify which service the client wishes to use (HTTP, telnet, ftp, etc.). Think of this like a telephone call: a telephone number is like an IP address and a port is an extension. Ports allow networked computers to provide many services but use only a single address.

Applications use standard port numbers to communicate. Some standard services and ports are:

FTP	20, 21
Telnet	23
SMTP (e-mail)	25
HTTP	80

When you type a URL into a Web browser to request a Web page via HTTP, it will try to connect to the server at port 80 unless you specify a different port number. There may be times when you want to run a service on a nonstandard port. For instance, you might have a production Web server running on port 80 but set up another HTTPD on port 8080 for testing purposes. On UNIX servers, port numbers below 1024 are available only for use by programs running as the root user (the system administrator). Ports above 1023 are available to programs running as any normal user provided that the port is not already in use. Once a daemon starts running on a port, any client can connect to it.

1.1.2 Describe Basic Functionality of Web Servers and Browsers

a) What is the primary function of an HTTP server?

Answer: The primary function of an HTTP server is to service client requests for documents. It waits for HTTP requests and then returns data for each one. An HTTP daemon provides an HTTP service. It allows a server to support client requests for documents. It generates errors when invalid requests are received or when a document cannot be found. The Web server process also generates log files of requests, errors, and other information.

b) Who developed the first Web server? What other early Web servers were developed?

Answer: The European Laboratory for Particle Physics (CERN) produced one of the first Web servers. The World Wide Web Consortium (W3C) took over development of the CERN HTTPD (also known as the W3C HTTPD), but no longer supports it. The W3C currently supports a Java-based server known as Jigsaw. Both the CERN HTTPD and Jigsaw are reference implementations, *meaning that they illustrate features of HTTP but are not meant for large-scale production use. Source code is available for both servers and they are excellent points of reference for developers wishing to write their own HTTP daemons.*

The National Center for Supercomputing Applications (NCSA) also created an HTTP server early in the evolution of the Internet. The CERN HTTPD was difficult to configure and not available for many platforms, so NCSA wrote their own version. The NCSA server quickly became the most popular Web server on the WWW from 1993 to 1995. Like the CERN server, however, development on the NCSA HTTPD has also ceased. Apache is a popular server based on the NCSA implementation. Originally written using existing code from the NCSA HTTPD, it has since been rewritten completely. Currently, Apache is the most widely used Web server software, with close to 50 percent market share.

Apache, CERN, and NCSA all released the source code for their Web servers. This made fixing bugs easier because anyone could see how the server worked. These servers make excellent examples for Web server developers, and they allow easy modification or customization of any aspect of the server.

c) What is the primary function of a Web browser?

Answer: The primary function of a Web browser is to display HTML documents. Although it can be used to view local documents on a hard drive, it is normally used as a client to retrieve documents from an HTTP server. Although browser software has expanded over the past few years to include such services as e-mail and news, its primary function is to format HTML documents for display.

d) What was the first Web browser? Why did it succeed where similar services (such as ftp, gopher, and WAIS) failed?

Answer: The first real HTML browser, NCSA Mosaic, came into being in early 1993. Although the hypertext documents had been around for some time, Mosaic had several essential features that made it popular right from the start. First, it was free, as are most browsers even today. Second, it was available for all major platforms: UNIX, Macintosh, and Microsoft Windows. Third, it was easy to create content—no special software was required to write HTML, only a text editor. Before Mosaic, only text-based clients such as gopher, WAIS, telnet, and FTP were widely available for retrieving information on the Internet. An easy-to-use GUI interface and easy-to-create content launched the Web in the form of NCSA Mosaic clients and HTTPD servers.

LAB 1.1 SELF-REVIEW QUESTIONS

To test your progress, you should be able to answer the following questions.

1) A Web server is which of the following?
- **a)** _____ Software
- **b)** _____ Hardware
- **c)** _____ Both a and b

2) A Web server can run on just about any type of machine, not just a huge, expensive server.
- **a)** _____ True
- **b)** _____ False

3) A browser utilizes which of the following technologies? (Choose all that apply.)
- **a)** _____ A network
- **b)** _____ A Web server
- **c)** _____ A phone line
- **d)** _____ HTTP

4) Which of the following may be a reason for running a Web server on a port other than port 80?

a) _____ You don't have access to port 80 (since you aren't root).
b) _____ You are running multiple Web servers on the same machine.
c) _____ You don't have enough memory.
d) _____ Both a and b
e) _____ All of the above

5) A server can also be a client.

a) _____ True
b) _____ False

Answers appear in Appendix A.

LAB 1.2

ELECTRONIC PUBLISHING

LAB OBJECTIVES

After completing this lab, you will be able to:

- Understand the Basics of Creating Hypertext Documents
- Understand the Difference between ASCII and Binary Files
- Give Examples of MIME Types

To understand more about Web servers and HTTP transactions, one must also be aware of how authors create and publish electronic documents. Although the focus of this book is not content creation, it is a good idea to familiarize yourself with some of the more technical aspects of electronic documents.

One of the strengths of the Web is the support of hypertext documents. A hypertext document contains hyperlinks (commonly referred to as *links*) that allow the reader to jump easily from one document to another, or to move around the current document. Links allow the user to follow a specific thread or view quickly documents on related topics. The Web is not limited to *text* documents, though; HTML documents can contain images, sounds, animations, and even video. Web publishing is about creating hypermedia, not just hypertext.

In the Web-publishing realm, we deal with two types of files: ASCII text files and binary files. ASCII files can be HTML or plain text or some other simple format. Most other files tend to be of the binary kind. A simple

text editor (notepad, emacs, vi) can create ASCII text files. You can create HTML documents by writing the HTML tags yourself with a text editor. Most Web authors will use a good text editor to do some of their authoring but supplement its use with a specialized HTML authoring package. Netscape Composer, Microsoft FrontPage, Macromedia Dreamweaver, and Adobe PageMill are some widely used HTML authoring packages.

ASCII TEXT FILES

Strictly speaking, an HTML document is just an ASCII text file. ASCII is the most common way of storing plain text on a computer. It uses numerical values (from 0 to 127) to represent letters, numbers, and other characters. Each byte of the file represents a specific character. For example, the letter "A" is represented by the number 65, the letter "B" by 66, and so on. For a list of all the ASCII values, see Appendix B.

ASCII text files are not compressed and can usually be viewed or edited by any simple text editor. Most operating systems can view and edit plain text files easily. Most use ASCII for representing text. Part of the appeal of HTML is that it is very easy to view the source code. This allows anyone to see how a certain effect was created.

BINARY FILES

A binary file is one that generally does not contain plain text in ASCII format. Images, sounds, and even compressed ASCII files are all binary files. To view them, an application must interpret the file. Word processors also create binary files—although they create text documents; the application saves the document in a binary format. Your word processor may be able to read and write ASCII files, too, but the files do not contain formatting information (fonts, margin settings, and the like). Any image or sound editing application also deals with binary files.

IMAGES

There are hundreds of file formats available for storing graphics and images. Web browsers typically support only a handful of image formats, however. The most common types of images are GIF and JPEG formats. Each of these formats has strengths and weaknesses. Both formats use compression to reduce the size of the file. GIF uses a lossless compression, meaning that it does not lose any of the image quality. JPEG images, on the other hand, use a lossy compression in which a relatively small file size is achieved with sacrifice to the image quality. GIF supports up to 256 colors, while JPEG images support millions of colors.

Table 1.1 ■ Image File Formats

GIF	JPEG	PNG
256 colors (8-bit)	16 million colors (24-bit)	16 million colors (24-bit)
Lossless compression	Lossy compression	Lossless compression
Transparency	No transparency	Transparency and opacity
Can be animated	No animation	No animation

Another format that is just recently gaining popularity in Web publishing is the PNG (portable network graphic) format. PNG images offer millions of colors, lossless compression, and other features that make them a good alternative to GIF images in many cases. Table 1.1 summarizes the differences in these image formats.

AUDIO

Most browsers have the ability to play sound files. This ability allows Web authors to include sound clips in their HTML documents. Audio files are embedded in a page to play automatically, or they can be used as links to be played when a user clicks on a link to the sound file. There are three sound formats commonly used on the Web, one corresponding to each of the three major platforms. Most current browsers with audio capabilities can support all three formats, so authors are free to choose which format to use and not worry too much about compatibility issues. Table 1.2 summarizes the differences in the three most common audio formats: WAV, AIFF, and AU.

MIME TYPES

The *multipurpose internet mail extensions* (MIME) are a set of rules that allow multimedia documents to be exchanged among many different computer systems. MIME was originally designed for sending attachments in e-mail, but it is also incorporated into HTTP. MIME uses media types and subtypes to describe the format of a file.

A Web server must determine the MIME type of a file before it sends it to the browser. To do this, it looks at the filename extension (suffix) and then tries to find that suffix in the MIME types database. Usually, this database is just a text file named `MIME.types` that contains a list of media types and their associated file extensions. It then sends the MIME type along with the document to the browser. The browser can use the MIME type to determine how it should display the document. Both the

Table 1.2 ■ Audio File Formats

WAV Files	AIFF Files	AU Files
Originated on Windows-based machine (introduced with Windows 3.0)	Originated on Macintosh (audio interchange file format)	Originated on Sun Microsystems workstations (UNIX)
8-kHz, 8-bit mono to 44-kHz, 16-bit stereo	8-kHz, 8-bit mono to 48-kHz, 16-bit stereo	8-kHz, 8-bit mono to 48-kHz, 16-bit stereo
Formally known as RIFF WAVE audio	Used for Red Book CD audio	The "original" Internet sound file format
Can be compressed or uncompressed	Not compressed; very pure format	Can be compressed or uncompressed

server and client must have a simple MIME types database. On the server it is usually a text file. On the client, each user may have its own MIME settings, either in a file or as part of the operating system configuration. Windows maintains file type associations in the registry, while UNIX typically uses text files. Maintaining a database for each user allows users to customize their tools to use different applications, depending on what type of file they're trying to view.

There are currently seven different media types in use: application, audio, image, message, multipart, text, and video. These media types provide a high-level description of the type of data sent. MIME also uses subtypes to further describe the actual data. For example, HTML is a text format, so it falls into the `text` media type. Its subtype is just `html`, so the MIME type for an HTML document would be `text/html`. A plain text document is described by `text/plain`. Images fall into the image category; `image/gif` describes a GIF image and `image/jpeg` describes a JPEG image file.

LAB 1.2 EXERCISES

1.2.1 UNDERSTAND THE BASICS OF CREATING HYPERTEXT DOCUMENTS

Use a text editor (not a word processor or publishing program) to create a simple HTML document with a hyperlink to the Prentice Hall Web site (`http://www.phptr.com/`).

a) What happens when you view your page in a browser?

b) Click on the hyperlink; it should display the Prentice Hall home page. View the source of the Prentice Hall home page. What do you see?

1.2.2 Understand the Difference between ASCII and Binary Files

a) Find an image on the Prentice Hall home page. Can you determine what type of image it is?

b) View the image by itself, then view the source of the image in the browser as you did with an HTML file. What do you see?

1.2.3 Give Examples of MIME Types

a) View any Web page from a browser. How can you determine what the MIME type of the document is?

b) How are MIME types used when requesting or receiving documents on the Web?

LAB 1.2 EXERCISE ANSWERS

1.2.1 UNDERSTAND THE BASICS OF CREATING HYPERTEXT DOCUMENTS

Use a text editor (not a word processor or publishing program) to create a simple HTML document with a hyperlink to the Prentice Hall Web site (`http://www.phptr.com/`).

a) What happens when you view your page in a browser?

Answer: If you created a valid HTML document and saved it with a `.html` *extension, it should look like a simple Web page—as you'd expect. If you saved it with a* `.txt` *extension (which is the default for many text editors), you might be looking at the source HTML in your browser rather than a formatted version. And, of course, if the HTML you entered is not valid, you might see some rather strange results in the browser.*

Here is a simple example HTML document:

```
<HTML>
<TITLE>My web Page</TITLE>

This is a simple web page.<BR>

<A HREF=http://www.phptr.com>Click Here for Prentice
Hall</A>
</HTML>
```

To create an HTML document, simply enter this text into a text editor and save it as `myfile.html`. In Windows, use notepad by clicking on the "Run" option in the Start menu, and entering "notepad" as the program to open. Enter the text to create your document and then save it as `myfile.html` in the directory of your choice. It is very important to save it with a `.html` extension; if you don't, the browser will not know that it is an HTML document.

b) Click on the hyperlink; it should display the Prentice Hall home page. View the source of the Prentice Hall home page. What do you see?

Answer: Viewing the source of any HTML document you find on the Web should show you the source code used to generate the document. For very complex pages, a lot of source is displayed, and it is often hard to read. For simpler pages, however, you can see exactly how the page is put together. This text that you are viewing is plain ASCII text with no special formatting.

Clicking the right mouse button in Netscape or Internet Explorer brings up a menu that allows you to view the source of the current document. The "View Info" option in Netscape gives you valuable information about the page also. If you right-click on an image or other object in a page, the menu displays different options.

1.2.2 Understand the Difference between ASCII and Binary Files

a) Find an image on the Prentice Hall home page. Can you determine what type of image it is?

Answer: If you can determine the name of the image file, you should be able to determine the type by the filename extension. To find the name of the image, you might try looking at the source code. The filenames for all images in the document should be in the `<IMG>` *tags.*

Another way to get more information about an image is to right-click on the image in the browser to bring up the options menu. For Netscape, click on "View Image." This displays the image by itself in the browser window. Now right-click on the image and select "View Info." This should display some information about the image, including its MIME type. In Internet Explorer, you can right-click on an image in an HTML document and select properties from the pop-up menu. This will also display the type, size, and other information about the image.

b) View the image by itself, then view the source of the image in the browser as you did with an HTML file. What do you see?

Answer: In Netscape, do a "View Image" as in Exercise 1.2.1. Now right-click on the image and select "View Source" or select "Page Source" from the view menu. You should see a page full of garbage characters. This is binary data. Unlike ASCII text files, images are not meant to be viewed in text mode.

1.2.3 GIVE EXAMPLES OF MIME TYPES

a) View any Web page from a browser. How can you determine what the MIME type of the document is?

Answer: In Netscape, you can get information about the page that you're currently viewing by selecting "Page Info" from the view menu. In Internet Explorer, you can right-click in the document and select "Properties" from the option menu. Along with the MIME type, you can view other information about the document that is provided from the HTTP headers. Netscape shows the last modified time, when the document expires, and whether or not the document is cached.

For HTML documents you should see that the MIME type is `text/html`. The media type is "text"—HTML is fundamentally text. The subtype is `html`, which further describes the type of text.

b) How are MIME types used when requesting or receiving documents on the Web?

Answer: A browser is able to specify what types of data it is capable of displaying, and it specifies this by using MIME types. When a server returns a document, it must tell the browser what type of data is being returned, and it also specifies this by using a standard MIME type.

LAB 1.2 SELF-REVIEW QUESTIONS

To test your progress, you should be able to answer the following questions.

1) An HTML file contains:
- **a)** _____ Text
- **b)** _____ Images
- **c)** _____ Both text and images
- **d)** _____ Binary data

2) Which of the following types of tools cannot be used to create hypertext documents?
- **a)** _____ A simple text editor
- **b)** _____ A word processing program
- **c)** _____ An automatic HTML generator
- **d)** _____ A graphics utility
- **e)** _____ All of these are capable of creating hypertext documents.

3) Mime types are important for which of the following reasons?
- **a)** _____ They allow the browser and server to communicate.
- **b)** _____ They tell applications what kinds of documents are being sent.
- **c)** _____ They speed the transmission of binary files.
- **d)** _____ FTP uses them to determine how to transfer files.

4) What is the MIME type of an HTML document?
- **a)** _____ html/text
- **b)** _____ HTML
- **c)** _____ text/html
- **d)** _____ text/plain

5) Why is a simple text editor useful to a webmaster?
- **a)** _____ It generates plain text files with no special characters.
- **b)** _____ Text editors are generally available on all platforms.
- **c)** _____ In many cases it's quicker than using a large application.
- **d)** _____ All of the above

Answers appear in Appendix A.

LAB 1.3

HTTP OVERVIEW

LAB OBJECTIVES

After completing this lab, you will be able to:

- Identify the Parts of an HTTP Transaction
- Identify HTTP Request Methods
- Identify HTTP Headers and Server Responses

HTTP TRANSACTIONS

As you learned in Lab 1.1, HTTP is a protocol that allows Web browsers to talk to servers and exchange information. HTTP provides a standard way of communicating between browsers and Web servers—so any browser can talk to any server, provided that they both conform to the HTTP specification. HTTP expects the client to initiate a request and the server to respond. Each request and response has three parts: the request or status line, the header fields, and the entity body.

FOR EXAMPLE

When you type a URL into your browser, it initiates an HTTP request to a Web server. That request has the following sections:

- *Request line.* This line contains a request method, the document location, and the protocol version.
- *Header section.* This series of lines contains HTTP headers that are used to pass other information about the request, and about the client itself, to the server. A blank line then separates the header section from the entity body.
- *Entity body.* This section contains other data to be passed to the server. There is usually information here only when a form is submitted.

If we typed `http://webmaster.merrimack.edu/simple.html` as a URL into Netscape, the browser would issue an HTTP request similar to the following:

```
GET /simple.html HTTP/1.0
User-Agent: Mozilla/4.5 [en] (X11; SunOS 5.5.1 sun4m)
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
```


There is a request line, followed by two HTTP headers and no entity body. The request line has three parts: a request method, the document location, and the protocol version. In this case the method is a GET method, the document requested is `simple.html`, and the protocol is HTTP version 1.0. The client also passes the User-Agent and Accept headers to the server.

The server then responds to the request in a similar fashion:

- *Status line.* This line contains the protocol version, a status code, and a reason phrase.
- *Header section.* This series of lines contains HTTP headers that are used to pass other information about the response, and about the server itself, to the client. A blank line then separates the header section from the entity body.
- *Entity body.* This section, if present, contains the document (or object) requested.

For the previous example, the server response might look something like this:

```
HTTP/1.1 200 OK
Date: Mon, 04 Jan 1999 00:33:10 GMT
Server: Apache/1.3.1 (Unix)
Last-Modified: Tue, 20 Oct 1998 21:00:39 GMT
Content-Length: 49
Content-Type: text/html

<HTML>
Welcome to the webmaster server...
</HTML>
```

There is a status line, followed by a header section containing five headers, and the entity body, which is a simple HTML document. Like the request line, the status line has three parts: the protocol version, a status code, and a reason phrase. In this case, the server is using HTTP version

1.1, and the HTTP response code is 200, which means that the client's request was successful and the server's response contains the data requested. The header section contains several headers that tell us a little bit about the server and the document returned in the entity body.

REQUEST METHODS

The request line of a client request contains an HTTP command called a *request method*. The server uses the method command to determine what to do with the request. There are currently several methods defined by the HTTP 1.1 standard, but only a few are widely supported by HTTP servers. The most widely used methods are GET, HEAD, and POST. Method commands should be in all-capital letters.

THE GET METHOD

The GET method is used to retrieve information from the server. It is most commonly used to retrieve documents from the Web server. Nothing is passed to the server in the entity body because this method is simply a request. The document returned by the server could be a static HTML document, output generated by a CGI program, or it could be an error generated by the server if something is wrong with the request. The previous example illustrates a GET method.

The GET method can pass information to the server (usually to a CGI program), but it must be included as part of the URL. To pass parameters as part of the URL, the URL must be followed by a question mark (?) and then the parameter pairs.

THE HEAD METHOD

The HEAD method is identical to the GET method except that the server does not return a document; it returns only the header section for the request. The HEAD method is useful for verifying that a document exists for checking links or to get information about the file type and modification time only.

THE POST METHOD

The POST method allows the server to receive data from the client. It is most commonly used to send the data in HTML forms to the server for processing. This method passes data to the server in the entity body of the request.

OTHER METHODS

The PUT method is becoming more widely supported. It is used for publishing documents to the Web server from a client. Many of the latest HTML authoring packages support posting documents to a Web server via the PUT method (more on this in Chapter 3). The DELETE method is used to remove a document from a Web server.

SERVER RESPONSES

After an HTTP server receives a request, it attempts to process the request. If a document is requested, the Web server will attempt to find the document and return it. If form information is passed to the server, the HTTPD passes that information to the appropriate resource for processing and returns any output. If the resource requested cannot be located, or if there is something wrong with the request itself, the server generates an error.

The server response, like the client request, has three parts: the status line, header fields, and the entity body. The status line contains three things: the protocol version, the status code, and a description phrase. The protocol should always be HTTP. The status code is a three-digit integer result code defined by the HTTP specification. The first digit of the status code represents the category of the response. There are currently five categories:

1) *Informational.* The request was received and is being processed.
2) *Success.* The client request was successful.
3) *Redirection.* The client request was not performed; further action must be taken by the client.
4) *Client error.* The client's request was incomplete or incorrect and cannot be fulfilled.
5) *Server error.* The request was not fulfilled, due to a server problem.

Here are some of the most common response codes.

INFORMATIONAL 1XX

100 Continue The initial part of the request has been received and the client should continue.

SUCCESSFUL 2XX

200 OK	This is probably the most common response; it means that the client's request was successful and the server's response contains the resource requested.
204 No Content	The request was successful but the response is empty. The client should not do anything when it receives this message.

REDIRECTION 3XX

301 Moved Permanently	The URL requested is no longer valid. The server should return the new location.
302 Found (Moved Temporarily)	The URL requested currently resides in a different location.
304 Not Modified	The client performed a conditional GET (If-Modified-Since header) and the document has not been modified. The entity body is not sent.

CLIENT ERROR 4XX

400 Bad Request	The server could not understand the request.
403 Forbidden	The client requested data that it did not have permission to access.
404 Not Found	The resource requested was not found on the server.

SERVER ERROR 5XX

500 Internal Server Error	Something unexpected happened on the server side. The most common reason for receiving this error is a problem with a server side program.

HTTP HEADERS

The HTTP header section is used to transfer information between the client and server. A header has a name and a value associated with it. There is one header per line and each line contains the header name followed by a colon, a space, and the value of the header name. Headers are used to transfer information about the client to the server, and vice versa. They are also used to transfer data related to the returned document,

cache parameters, cookies, and other session information. Some of the most common HTTP headers are described below.

CLIENT REQUEST HEADERS

Accept	Used to specify which media types the client prefers to accept.
Cookie	Contains cookie information (name/value pair, etc.) for the URL requested.
If-Modified-Since	Used to do a *conditional GET* request. The server will return the document only if it has been modified since the date specified.
Referer	Allows the client to specify the URL of the page from which the currently requested URL was obtained.
User-Agent	Contains information about the client program originating the request. It is used to identify the browser software.

SERVER RESPONSE HEADERS

Server	Contains information about the server software handling the request.
Set-Cookie	Allows the server to set a cookie on the client browser (if permitted) for the given URL or domain.

ENTITY HEADERS

Content-Length	Specifies the size (in bytes) of the data transferred in the entity body. This header is sent for most static documents, but not for dynamically generated content (i.e., CGI programs).
Content-Type	Specifies the MIME type of the data returned in the entity body.
Expires	Specifies the time/date after which the response is considered outdated. This header is useful for caching documents—if the browser knows when the document will change, it does not need to retrieve a fresh copy until then.
Last-Modified	Specifies the date and time the document was last modified.

LAB 1.3 EXERCISES

1.3.1 IDENTIFY THE PARTS OF AN HTTP TRANSACTION

a) What are the three parts of every HTTP transaction?

1.3.2 IDENTIFY HTTP REQUEST METHODS

a) Name the three most widely used request methods.

b) What is the difference between a GET and a POST method?

c) What is the difference between a HEAD and a GET method?

1.3.3 IDENTIFY HTTP HEADERS AND SERVER RESPONSES

a) What header is sent by the client to identify and give information about the browser?

b) What header is sent by the server so that the browser can determine what type of content is being returned?

c) What header is sent by the server to identify the server software?

LAB 1.3 EXERCISE ANSWERS

1.3.1 IDENTIFY THE PARTS OF AN HTTP TRANSACTION

a) What are the three parts of every HTTP transaction?

Answer: A request or response line, a header section, and an entity body.

A request line is sent as the first line of all HTTP requests. The browser then sends any relevant headers. An entity body is sent only when data other than the headers needs to be sent to the server. A GET method does not usually contain an entity body, but a POST or PUT method usually does.

A response line is sent as the first line of all HTTP responses. The server then sends any relevant headers and the entity body. The entity body is usually the document requested, but it could also be error information if an error occurred while trying to retrieve the document.

1.3.2 IDENTIFY HTTP REQUEST METHODS

a) Name the three most widely used request methods.

Answer: GET, POST, and HEAD.

Currently, these are the most widely used request methods. As new features are added to the HTTP specification, other methods may become more widely used.

b) What is the difference between a GET and a POST method?

Answer: The GET method contains no entity body. To pass data to the server it must include the data in the URL. The POST method transfers data in the entity body.

c) What is the difference between a HEAD and a GET method?

Answer: The HEAD method is used to return the header section for a specific document; it does not return the document itself.

1.3.3 IDENTIFY HTTP HEADERS AND SERVER RESPONSES

a) What header is sent by the client to identify and give information about the browser?

Answer: The User-Agent request header contains information about the client program originating the request. This is not a required header, but most browsers send it when making a request. The server can use this header to determine what browser is requesting a document and to tailor its response if necessary.

Netscape sends a User-Agent header similar to the following:

```
Mozilla/4.5 [en] (X11; U; SunOS 5.5.1 sun4m)
```

Internet Explorer sends a User-Agent header similar to the following:

```
Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
```

b) What header is sent by the server so that the browser can determine what type of content is being returned?

Answer: The Content-Type header indicates the media type of the data contained in the entity body. The server determines the type of data by looking at the file extension and referencing the MIME types file.

c) What header is sent by the server to identify the server software?

Answer: The Server header field contains information about the HTTPD software.

The Apache Web server returns a Server header similar to the following:

```
Server: Apache/1.3.1 (Unix)
```

Microsoft's IIS returns a Server header similar to the following:

```
Server: Microsoft-IIS/4.0
```

LAB 1.3 SELF-REVIEW QUESTIONS

To test your progress, you should be able to answer the following questions.

1) What is the first thing that is passed to the server when an HTTP transaction begins?

a) _____ The request line
b) _____ The entity body
c) _____ The transaction line
d) _____ The header section

2) The GET method is the only method that retrieves information from the server.

a) _____ True
b) _____ False

3) What is the Referer header used for?

a) _____ It refers people to your site.
b) _____ It redirects URLs that no longer exist.
c) _____ It shows the link that was clicked to get to the page being requested.
d) _____ It is not used.

4) Headers are used by the browser to determine when a document will expire.

a) _____ True
b) _____ False

Answers appear in Appendix A.

LAB 1.4

OTHER WEB-RELATED SERVERS

LAB OBJECTIVES

After completing this lab, you will be able to:

- Understand the Functionality of Proxy Servers
- Identify Other Services That May Run Alongside an HTTP Server

A server that can communicate by HTTP is a great thing because it is able to communicate with millions of other computers. Any browser can retrieve your pages and view them. For many people, a system running an HTTP server suits their needs just fine, but there are other servers that you should know about. In this lab we discuss a few of the most common servers that run alongside an HTTP server.

PROXY SERVERS

A proxy server is an intermediary server that goes between a client and the destination server—a middleman. A browser configured to use a proxy server for all requests allows the proxy server to process the request and response. Instead of connecting directly to the destination server when a request for a URL is made, the browser sends the request to the proxy. The proxy then passes the request to the destination server, receives the response, and passes the response back to the browser. This may sound like a lot of work, but having a proxy machine in the middle of the transaction allows some extra processing of the returned data to take place.

Proxy servers have three main uses: security, content filtering, and caching. Used for security purposes, the proxy can act as a firewall, allow-

ing only HTTP traffic through and rejecting other protocols. A firewall limits what kinds of services are available to people outside your local network. You might only want to allow HTTP requests to get to your server and deny FTP, telnet, and other services. Proxies can also filter data, restricting access to certain sites or analyzing content for questionable material. Caching proxy servers help improve performance by storing frequently accessed documents locally.

Security uses of proxy servers are covered in detail in Chapter 10, so let's take a look at the other two uses for proxies: filtering and caching. Restricting access to content based on file type is another possible use for a proxy. In terms of security, html documents are reasonably harmless, but executable files can pose a threat to security. A system administrator may choose to allow only nonexecutable content through the proxy, blocking `.exe` files and similar documents that execute on local hosts. By allowing only simple text documents and images through the proxy, it is much more difficult for viruses and hackers to gain access to computers on your side of the proxy.

Not all Web pages are cacheable, because content is dynamically generated. HTTP headers play a big role in determining if a new document needs to be retrieved or if the cached document is still valid. The Expires HTTP header specifies when the document may change. A Web cache can look at this header to determine if the document is still valid. If the Expires header is set to a time in the past, a new document is retrieved; otherwise, the cached version is returned. If the server did not set an Expires header, the client can use the If-Modified-Since header to fetch the document only if it had not been modified since a certain date. The client requests the document conditionally with a GET method, and the server returns the document or issues a 304—Not Modified response code if the document has not changed.

Your browser software must be explicitly configured to use a proxy server. Figure 1.1 shows a sample configuration dialog from Netscape Navigator. We use MediaOne Express as our Internet service provider (ISP) at home and they provide a proxy cache server for their users. The cache server stores frequently accessed Web pages so when one is requested, the cache server can return the page rather than retrieving it from a distant server on the Internet. Using the proxy server makes pages that we go to load much quicker. It also makes better use of the ISP's bandwidth by going outside the local network only when new pages need to be retrieved.

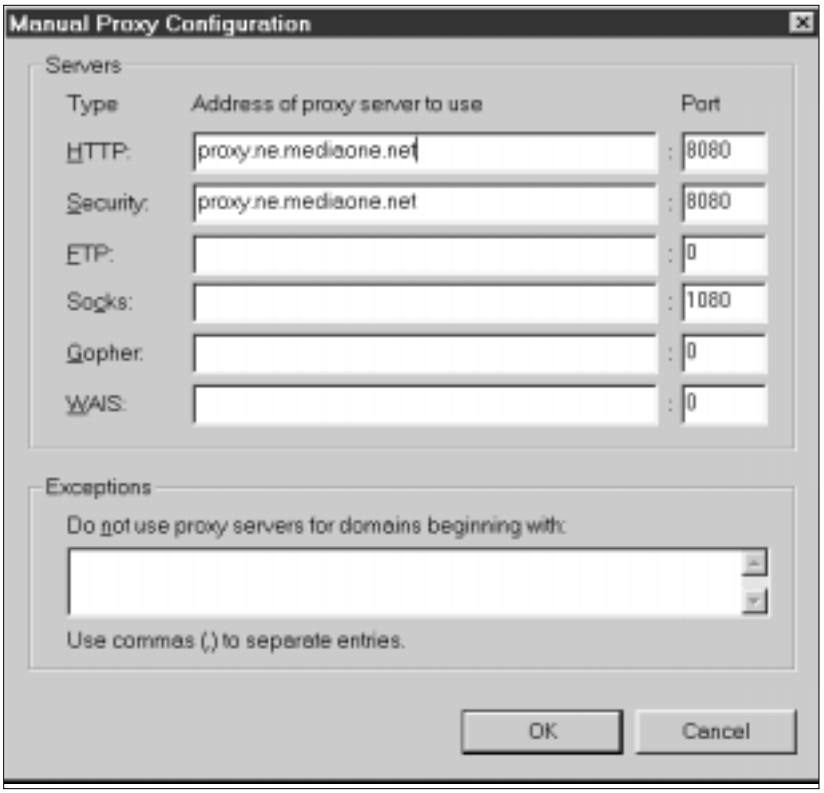

Figure 1.1 ■ Netscape Navigator Proxy Settings

■ *FOR EXAMPLE*

To configure your browser to use a proxy server, you must first obtain the names of your local proxies or a URL that has the correct proxy configuration information. Your ISP might provide proxy servers, but not all do. For Netscape:

- Click on “Preferences” in the Edit menu.
- Select “Proxies” from the Advanced tab.
- The default is a direct connection to the Internet—no proxies. Clicking on “Manual Proxy Configuration” allows you to select “View,” which brings up the dialog shown in Figure 1.1. Automatic Proxy configuration allows you to specify a URL containing proxy information. This allows the system administrator to change proxies dynamically.

If you don't have a proxy server already, you can set one up yourself. Many of the Web server packages discussed in Lab 4.1 offer proxy services in addition to normal HTTP server capabilities.

STREAMING AUDIO AND VIDEO

For a browser to play an audio or video file, it must first download the entire file. Over a modem connection, it takes a long time to download a few minutes of audio or a few seconds of video. The solution: streaming media, which allow a media player (or plug-in) to start playing multimedia content while the data is still being received. Instead of having to wait for the entire file to download, the player can start almost immediately. A streaming media server can broadcast live audio/video feeds (from a video capture card, for instance) or serve prerecorded clips.

HTTP does not support streaming media, so a different server must be used to publish streaming media. Browsers don't support streaming media, so a plug-in must be used to view any type of streaming content. When a user clicks on a link for a streamed file, the browser will start up the appropriate player. That player will connect to the server at a specific port and request a file or live stream, much like an HTTP transaction. As the player starts receiving the data, it may store a few seconds' worth in a buffer and then start to play the stream—whether audio, video, or both. With traditional audio and video files, the entire file must be downloaded before a player can read it. In addition, unlike HTTP, many streaming media formats may use UDP instead of TCP/IP as a network protocol.

UDP is good at transmitting very small pieces of data quickly, and for digital audio and video, it works quite well. Unlike TCP/IP, UDP will not retransmit data if there is an error. This is fine for digital audio and video because a few bits lost here or there will hardly be noticeable. Lost or delayed data may account for pops and clicks in audio as it plays back. While TCP/IP offers reliability, it is somewhat slower than UDP, so it is used primarily when a UDP connection is unavailable for some reason.

The two leading streaming media packages are RealNetworks' RealSystem and Microsoft's Windows Media (formerly NetShow). Both packages offer similar features and quality.

FTP

FTP (File Transfer Protocol) is used to transfer files between computers on a network. A host with a Web server running on it may also set up an FTP server so that Web pages can be uploaded to the server easily. Like HTTP,

FTP relies on client and server software. The FTP daemon (FTPD) is a program that runs on the server and allows clients to connect. It provides a means of authentication so that only authorized users can transfer files to and from the server. UNIX servers generally install an FTPD by default, and an FTP server can be installed on Windows NT along with Microsoft's IIS (Internet Information Server). FTP clients are available for just about any operating system. UNIX and Microsoft Windows both come with a simple, text-based FTP client that can be used to transfer files to any server running an FTPD. Although FTP is not the only way to transfer files to a server, it is one of the most widely supported.

DATABASES

Most business sites rely on some sort of database, either for E-commerce transaction processing or to allow access to current support documents or product information, for example. A database provides an efficient, organized way to store lots of information. Unfortunately, most databases don't provide a friendly interface that anyone can use to access this information. The Web provides a familiar, easy-to-use way of accessing data, and a Web developer can easily write programs that run on the Web server and display information from a database.

A large corporate database should typically be installed on its own dedicated server and not on a machine also used as a Web server. A large database requires lots of memory, disk space, and CPU power, so installing it on a machine that is also trying to process Web pages may be a bad idea. The database will also have a daemon running to respond to queries; this allows programs on the Web server to communicate with the database server. This type of database daemon is often called a *listener*. Many database packages now come with tools to make authoring Web-database applications much easier. Products such as Oracle 8I include HTML generation tools, integration with Java, and may even provide a Web server built into the database software.

SSL

By default, HTTP traffic is transmitted in clear text; it is not encrypted. This is fine for most general surfing, but if you want to start sending confidential information over the Web, it becomes an issue. Secure Sockets Layer (SSL) is a protocol that allows secure, encrypted communication over TCP/IP. It is often used with HTTP to allow information to be exchanged securely between a browser and a Web server. Most commercial Web server software includes an SSL server that can run alongside the http daemon. SSL is used mostly for Web transactions, but it can be used to encrypt any communications over TCP/IP. Netscape developed the SSL

standard that is now supported by most browsers. SSL is covered in more detail in Lab 4.5 and Chapter 13.

LAB 1.4 EXERCISES

1.4.1 UNDERSTAND THE FUNCTIONALITY OF PROXY SERVERS

a) What are the benefits of a caching proxy server?

b) How is a proxy used to filter content?

c) Explain what happens when a URL is requested by a browser that is configured to use a proxy.

1.4.2 IDENTIFY OTHER SERVICES THAT MAY RUN ALONGSIDE AN HTTP SERVER

a) Why is a streaming audio server useful if you want to deliver audio content?

b) Why is an FTP server useful on a machine running a Web server?

LAB 1.4 EXERCISE ANSWERS

1.4.1 UNDERSTAND THE FUNCTIONALITY OF PROXY SERVERS

a) What are the benefits of a caching proxy server?

Answer: A caching proxy server helps improve performance for intranets. In general terms, a cache is something that keeps frequently used data available for quick access. When a user requests a URL, the proxy server checks to see if it has a local copy. If it does, that copy may get returned rather than fetching the document from the real Web site again. This is similar to the disk cache that Web browsers use, but by keeping all the documents on a local server, many users are able to benefit. The speed of intranets is typically very fast compared to the connection to the Internet, so retrieving files from a local server is noticeably faster than retrieving files from an external server somewhere on the Internet.

To improve performance without using a dedicated proxy cache server, your Web browser most likely has a local disk cache that holds many of the Web pages you have recently viewed. If you go back to one of those pages, your browser will just grab the file off the hard drive rather than retrieve the same document from the Web server. This saves network bandwidth and makes browsing much faster on your end.

b) How is a proxy used to filter content?

Answer: There are many reasons to filter Web content; the most common is to deny access to certain pages. Many schools set up proxy servers to filter "inappropriate" content. When the proxy receives a page that contains certain words that are deemed unsuitable, instead of returning the page to the browser, it will return a page saying that the page requested cannot be viewed. This method restricts access on a per-page basis. Some pages at a given site are viewable, while others that contain questionable material are blocked. Another method of restricting access is on a per-site basis. The proxy can be configured to block access to entire Web sites that are considered unacceptable.

c) Explain what happens when a URL is requested by a browser that is configured to use a proxy.

Answer: The browser will actually make an HTTP connection to the proxy server, not the Web server requested. The proxy receives the request from the browser and then makes a connection to the Web server for that URL. The proxy server retrieves the response and then returns the data to the client requesting it.

1.4.2 Identify Other Services That May Run Alongside an HTTP Server

a) Why is a streaming audio server useful if you want to deliver audio content?

Answer: Streaming audio allows users to listen to long audio clips (or even live audio feeds) without having to wait for a large audio file to download. The client will start playing the audio almost instantly. This works well as long as the network is fast enough to support the constant flow of data. Compression algorithms make the audio data small enough that even a modem connection is fast enough for decent-sounding audio transmission.

b) Why is an FTP server useful on a machine running a Web server?

Answer: FTP provides an easy, standard way of transferring files to a Web server. FTP clients are available for many platforms, so just about anyone can use it. Since Web pages may be created on other machines, there must be a way to publish those files to the server. FTP provides some security by requiring a login and password, although anonymous logins are possible. Most server operating systems provide an FTP daemon as part of the core OS.

LAB 1.4 SELF-REVIEW QUESTIONS

To test your progress, you should be able to answer the following questions.

1) Proxy servers are required to allow an intranet to access the Internet.
- **a)** _____ True
- **b)** _____ False

2) A proxy server can be used for caching or filtering, but not both.
- **a)** _____ True
- **b)** _____ False

3) Which of the following is not used to transfer files to a Web server?
- **a)** _____ MS FrontPage server extensions
- **b)** _____ FTP
- **c)** _____ Telnet
- **d)** _____ A modem

4) Which of the following is not a function of a proxy server?
- **a)** _____ Security
- **b)** _____ CGI programming
- **c)** _____ Caching
- **d)** _____ Filtering

Answers appear in Appendix A.

CHAPTER 1

TEST YOUR THINKING

The projects in this section use the skills you've acquired in this chapter. The answers to these projects are available to instructors only through a Prentice Hall sales representative and are intended to be used in classroom discussion and assessment.

1) Create a simple HTML document on your local system and view it with your favorite browser.

   a) Try changing the extension from .html to .txt and view it in your browser. Is anything different?

2) Upload the document to a server using FTP. Open the correct URL up with a browser.

3) Connect to the server at port 80 with a telnet client; issue a GET command to retrieve the file.