Application Layer

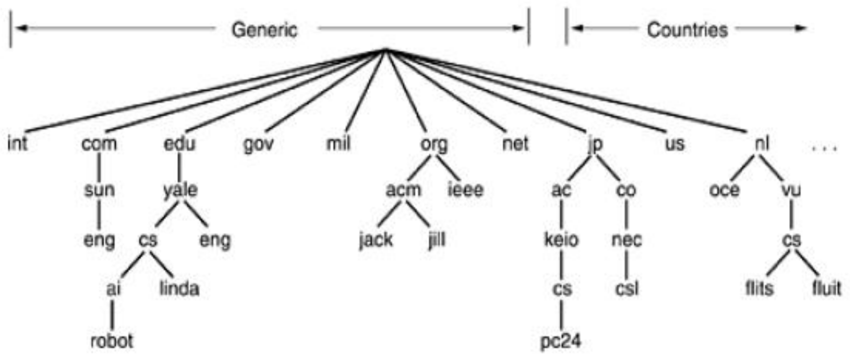
Unit 6

DNS—The Domain Name System

Although programs theoretically could refer to hosts, mailboxes, and other resources by their network (e.g., IP) addresses, these addresses are hard for people to remember. Also, sending e-mail to *tana@128.111.24.41* means that if Tana's ISP or organization moves the mail server to a different machine with a different IP address, her e-mail address has to change. Consequently, ASCII names were introduced to decouple machine names from machine addresses. In this way, Tana's address might be something like *tana@art.ucsb.edu*. Nevertheless, the network itself understands only numerical addresses, so some mechanism is required to convert the ASCII strings to network addresses. In the following sections we will study how this mapping is accomplished in the Internet.

Way back in the ARPANET, there was simply a file, *hosts.txt*, that listed all the hosts and their IP addresses. Every night, all the hosts would fetch it from the site at which it was maintained. For a network of a few hundred large timesharing machines, this approach worked reasonably well.

However, when thousands of minicomputers and PCs were connected to the net, everyone realized that this approach could not continue to work forever. For one thing, the size of the file would become too large. However, even more important, host name conflicts would occur constantly unless names were centrally managed, something unthinkable in a huge international network due to the load and latency. To solve these problems, **DNS** (the **Domain Name System**) was invented.



Electronic Mail

Electronic mail, or **e-mail**, as it is known to its many fans, has been around for over two decades. Before 1990, it was mostly used in academia. During the 1990s, it became known to the public at large and grew exponentially to the point where the number of e-mails sent per day now is vastly more than the number of **snail mail** (i.e., paper) letters.

E-mail, like most other forms of communication, has its own conventions and styles. In particular, it is very informal and has a low threshold of use. People who would never dream of calling up or even writing a letter to a Very Important Person do not hesitate for a second to send a sloppily-written e-mail.

E-mail is full of jargon such as BTW (By The Way), ROTFL (Rolling On The Floor Laughing), and IMHO (In My Humble Opinion). Many people also use little ASCII symbols called **smileys** or **emoticons** in their e-mail. A few of the more interesting ones are reproduced in Fig. 7-6. For most, rotating the book 90 degrees clockwise will make them clearer.

The World Wide Web

The World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet. In 10 years, it went from being a way to distribute high-energy physics data to the application that millions of people think of as being "The Internet." Its enormous popularity stems from the fact that it has a colorful graphical interface that is easy for beginners to use, and it provides an enormous wealth of information on almost every conceivable subject, from aardvarks to Zulus.

The Web (also known as **WWW**) began in 1989 at CERN, the European center for nuclear research. CERN has several accelerators at which large teams of scientists from the participating European countries carry out research in particle physics. These teams often have members from half a dozen or more countries. Most experiments are highly complex and require years of advance planning and equipment construction. The Web grew out of the need to have these large teams of internationally dispersed researchers collaborate using a constantly changing collection of reports, blueprints, drawings, photos, and other documents.

7.3.1 Architectural Overview

From the users' point of view, the Web consists of a vast, worldwide collection of documents or **Web pages**, often just called **pages** for short. Each page may contain links to other pages anywhere in the world. Users can follow a link by clicking on it, which then takes them to the page pointed to. This process can be repeated indefinitely. The idea of having one page point to another, now called **hypertext**, was invented by a visionary M.I.T. professor of electrical engineering, Vannevar Bush, in 1945, long before the Internet was invented.

Pages are viewed with a program called a **browser**, of which Internet Explorer and Netscape Navigator are two popular ones. The browser fetches the page requested, interprets the text and formatting commands on it, and displays the page, properly formatted, on the screen. An example is given in Fig. 7-18(a). Like many Web pages, this one starts with a title, contains some information, and ends with the e-mail address of the page's maintainer. Strings of text that are links to other pages, called **hyperlinks**, are often highlighted, by underlining, displaying them in a special color, or both. To follow a link, the user places the mouse cursor on the highlighted area, which causes the cursor to change, and clicks on it. Although nongraphical browsers, such as Lynx, exist, they are not as popular as graphical browsers, so we will concentrate on the latter. Voice-based browsers are also being developed.

HTTP—The HyperText Transfer Protocol

The transfer protocol used throughout the World Wide Web is **HTTP** (**HyperText Transfer Protocol**). It specifies what messages clients may send to servers and what responses they get back in return. Each interaction consists of one ASCII request, followed by one RFC 822 MIME-like response. All clients and all servers must obey this protocol. It is defined in RFC 2616. In this section we will look at some of its more important properties.

Connections

The usual way for a browser to contact a server is to establish a TCP connection to port 80 on the server's machine, although this procedure is not formally required. The value of using TCP is that neither browsers nor servers have to worry about lost messages, duplicate messages, long messages, or acknowledgements. All of these matters are handled by the TCP implementation.

In HTTP 1.0, after the connection was established, a single request was sent over and a single response was sent back. Then the TCP connection was released. In a world in which the typical Web page consisted entirely of HTML text, this method was adequate. Within a few years, the average Web page contained large numbers of icons, images, and other eye candy, so establishing a TCP connection to transport a single icon became a very expensive way to operate.

This observation led to HTTP 1.1, which supports **persistent connections**. With them, it is possible to establish a TCP connection, send a request and get a response, and then send additional requests and get additional responses. By amortizing the TCP setup and release over multiple requests, the relative overhead due to TCP is much less per request. It is also possible to pipeline requests, that is, send request 2 before the response to request 1 has arrived.

Methods

Although HTTP was designed for use in the Web, it has been intentionally made more general than necessary with an eye to future object-oriented applications. For this reason, operations, called **methods**, other than just requesting a Web page are supported. This generality is what permitted SOAP to come into existence. Each request consists of one or more lines of ASCII text, with the first word on the first line being the name of the method requested. The built-in methods are listed in Fig. 7-41. For accessing general objects, additional object-specific methods may also be available. The names are case sensitive, so *GET* is a legal method but *get* is not.

Message Headers

The request line (e.g., the line with the *GET* method) may be followed by additional lines with more information. They are called **request headers**. This information can be compared to the parameters of a procedure call. Responses may also have **response headers**. Some headers can be used in either direction.

Security is a broad topic and covers a multitude of sins. In its simplest form, it is concerned with making sure that nosy people cannot read, or worse yet, secretly modify messages intended for other recipients. It is concerned with people trying to access remote services that they are not authorized to use. It also deals with ways to tell whether that message purportedly from the IRS saying: Pay by Friday or else is really from the IRS and not from the Mafia. Security also deals with the problems of legitimate messages being captured and replayed, and with people trying to deny that they sent certain messages.

Most security problems are intentionally caused by malicious people trying to gain some benefit, get attention, or to harm someone. A few of the most common perpetrators are listed in Fig. 8-1. It should be clear from this list that making a network secure involves a lot more than just keeping it free of programming errors. It involves outsmarting often intelligent, dedicated, and sometimes well-funded adversaries. It should also be clear that measures that will thwart casual adversaries will have little impact on the serious ones. Police records show that most attacks are not perpetrated by outsiders tapping a phone line but by insiders with a grudge. Consequently, security systems should be designed with this fact in mind.

Introduction to Cryptography

Historically, four groups of people have used and contributed to the art of cryptography: the military, the diplomatic corps, diarists, and lovers. Of these, the military has had the most important role and has shaped the field over the centuries. Within military organizations, the messages to be encrypted have traditionally been given to poorly-paid, low-level code clerks for encryption and transmission. The sheer volume of messages prevented this work from being done by a few elite specialists.

Until the advent of computers, one of the main constraints on cryptography had been the ability of the code clerk to perform the necessary transformations, often on a battlefield with

little equipment. An additional constraint has been the difficulty in switching over quickly from one cryptographic method to another one, since this entails retraining a large number of people. However, the danger of a code clerk being captured by the enemy has made it essential to be able to change the cryptographic method instantly if need be.

The messages to be encrypted, known as the **plaintext**, are transformed by a function that is parameterized by a **key**. The output of the encryption process, known as the **ciphertext**, is then transmitted, often by messenger or radio. We assume that the enemy, or **intruder**, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know what the decryption key is and so cannot decrypt the ciphertext easily. Sometimes the intruder can not only listen to the communication channel (passive intruder) but can also record messages and play them back later, inject his own messages, or modify legitimate messages before they get to the receiver (active intruder). The art of breaking ciphers, called **cryptanalysis**, and the art devising them (cryptography) is collectively known as **cryptology**.