APPLICATION LAYER PARADIGMS

The layer where all the applications are found is called Application layer.The traditional paradigm is called the client-server paradigm. In this paradigm, the service provider is an application program, called the server process; it runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.

Normally few server processes are available that can provide a specific type of service, but there are many clients that request service from any of these server processes.The server process must be running all the time; the client process is started when the client needs to receive service.

For example, a telephone directory center in any area can be a server; a subscriber that calls and asks for a specific telephone number can be thought of as a client.

The directory center must be ready and available all the time; the subscriber can call the center for a short period when the service is needed. Figure5.1.1 shows an example of a client-server communication in which three clients communicate with one server to receive the services provided by this server.

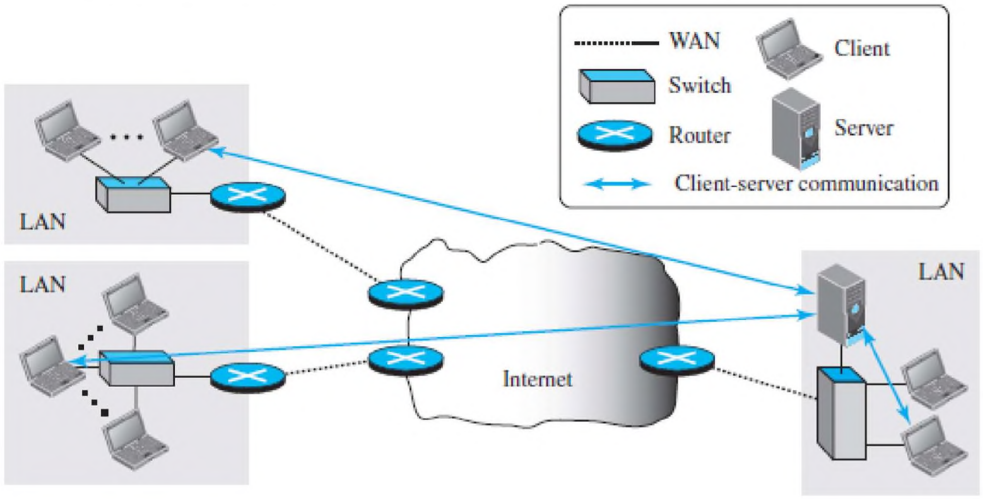

Fig5.1.1:Client server paradigm.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-821]

Peer-to-Peer paradigm

In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect.The responsibility is shared between peers. A computer connected

to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time.Figure 5.1.2 shows an example of communication in this paradigm.

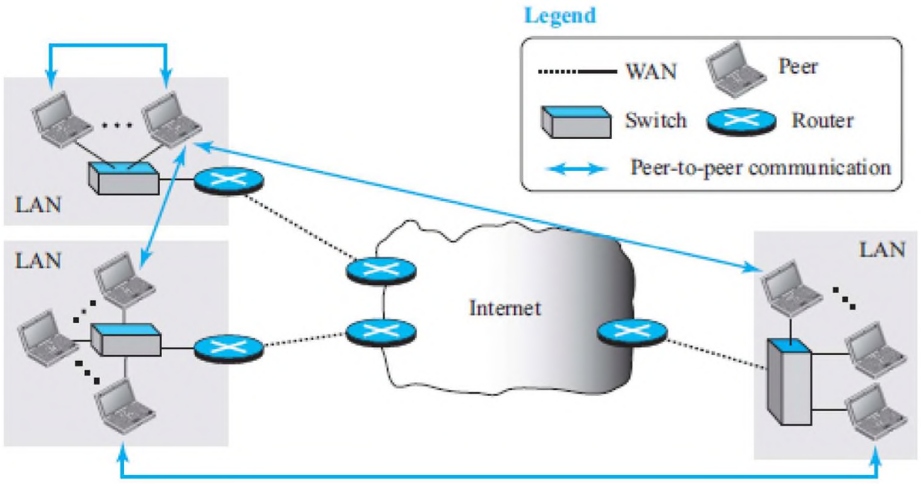

Fig5.1.2: Peer to peer paradigm.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-822]

Communication by phone is a peer-to-peer activity; no party needs to wait for the other party to call.The peer-to-peer paradigm can be used in a situation, when some computers connected to the Internet have something to share with each other.For example, if an Internet user has a file available to share with other Internet users, there is no need for the file holder to become a server and run a server process all the time waiting for other users to connect and to get the file.

Client-Server Programming

In a client-server paradigm, communication at the application layer is between two running application programs called processes: a client and a server. A client is a running program that initializes the communication by sending a request; a server is another application program that waits for a request from a client.The server handles the request received from a client, prepares a result, and sends the result back to the client.

The lifetime of a server is infinite: it should be started and run forever, waiting for the clients. The lifetime of a client is finite. It sends a finite number of requests to the corresponding server, receives the responses, and stops.

Application Programming Interface

If we need a process to be able to communicate with another process, we need a new set of instructions to tell the lowest four layers of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection.A set of instructions of this type is called as an application programming interface (API).An interface in programming is a set of instructions between two entities. In this case, one of the entities is the process at the application layer and the other is the operating system that encapsulates the first four layers of the TCP/IP protocol suite.

A computer manufacturer build the first four layers of the suite in the operating system and include an API. Here, the processes running at the application layer are able to communicate with the operating system when sending and receiving messages through the Internet.

Several APIs have been designed for communication. Three are common: socket interface, Transport Layer Interface (TLI), and STREAM.

Socket Interface

Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment.

The socket interface is a set of instructions that provide communication between the application layer and the operating system. It is a set of instructions that can be used by a process to communicate with another process. For example, in most computer languages, like C, C++, or Java, we have several instructions that can read and write data to other sources and sinks such as a keyboard (a source), a monitor (a sink), or a file (source and sink). Figure 5.1.3 shows the socket format.

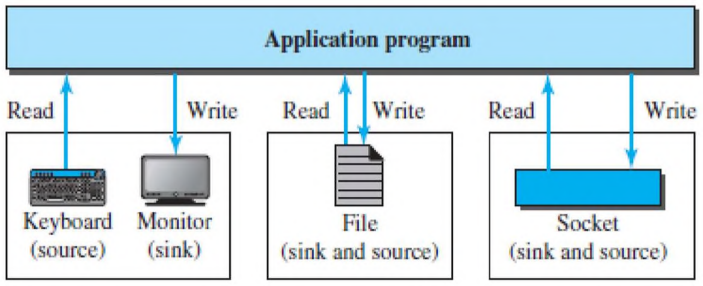

Fig5.1.3: Socket format.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-824]

In application layer, communication between a client process and a server process is the communication between two sockets.As far as the application layer is concerned, communication between a client process and a server process is communication between two sockets, created at two ends, as shown in Figure 5.1.4. The client thinks that the socket is the entity that receives the request

and gives the response; the server thinks that the socket is the one that has a request and needs the response. If we create two sockets, one at each end, and define the source and destination addresses correctly, we can use the available instructions to send and receive data. The rest is the responsibility of the operating system and the embedded TCP/IP protocol.

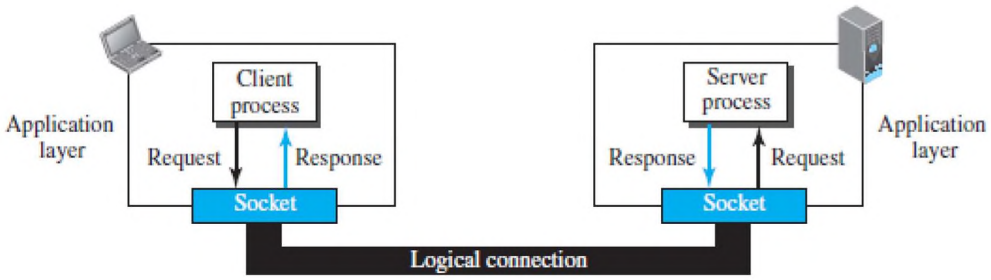

Fig5.1.4: Socket in process to process communication.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-825]

Socket Addresses

The interaction between a client and a server is two-way communication. In a two-way communication, we need a pair of addresses: local (sender) and remote (receiver). The local address in one direction is the remote address in the other direction and vice versa.Since communication in the client-server paradigm is between two sockets, we need a pair of socket addresses for communication: a local socket address and a remote socket address.A socket address should first define the computer on which a client or a server is running. A computer in the Internet is defined by its IP address. a socket address should be a combination of an IP address and a port number as shown in Figure 5.1.5.

Fig5.1.5: A Socket address.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-825]

Finding Socket Addresses

To find socket address, the situation is different for each site.

Local Socket Address The local (server) socket address is provided by the operating system. The operating system knows the IP address of the computer on which the server process is running.

The port number of a server process, however, needs to be assigned. If the server process is a standard one defined by the Internet authority, a port number is already assigned to it. For example, the assigned port number for a Hypertext Transfer Protocol (HTTP) is the integer 80, which cannot be used by any other process.

If the server process is not standard, the designer of the server process can choose a port number, in the range defined by the Internet authority, and assign it to the process. When a server starts running, it knows the local socket address.

Remote Socket Address

The remote socket address for a server is the socket address of the client that makes the connection. Since the server can serve many clients, it does not know previously, the remote socket address for communication.The server can find this socket address when a client tries to connect to the server. The client socket address, which is contained in the request packet sent to the server, becomes the remote socket address that is used for responding to the client.

WORLD WIDE WEB

The Web was first proposed by Tim Berners-Lee in 1989 at *CERN*†, the European Organization for Nuclear Research, to allow several researchers at different locations throughout Europe to access each others researches.The commercial Web started in 1990s.The web pages,are distributed all over the world and related documents are linked together.Today, the Web is used to provide electronic shopping and gaming.

Architecture of WWW

The WWW is a distributed client-server service, in which a client using a browser can access a service using a server. The service provided is distributed over many locations called sites as in figure 5.2.1.

A web page can be simple or composite. A simple web page has no links to other web pages; a composite web page has one or more links to other web pages. Each web page is a file with a name and address.

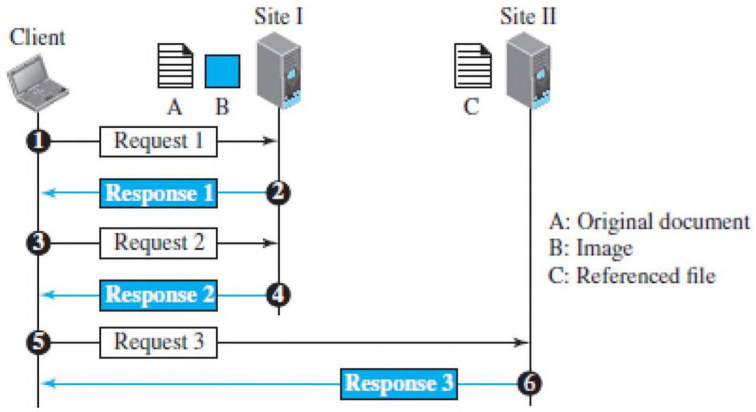

Fig5.2.1: The web architecture.

[Source :"Data Communications and Networking" by Behrouz A. Forouzan,Page-821]

The first transaction (request/response) retrieves a copy of the main document (file A), which has references (pointers) to the second and third files. When a copy of the main document is retrieved and browsed, the user can click on the reference to the image to invoke the second transaction and retrieve a copy of the image (file B). If the user needs to see the contents of the referenced text file, she can click on its reference (pointer) invoking the third transaction and retrieving a copy of file C.

Web Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a web page, and all of them use the same architecture.Each browser has three parts: a controller, client protocols, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document.

After the document has been accessed, the controller uses one of the interpreters to display the document on the screen.The client protocol can be one of the protocols described later, such as HTTP or FTP.The interpreter can be HTML, Java, or JavaScript, depending on the type of document. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

Web Server

The web page is stored at the server. Each time a request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than a disk.

Uniform Resource Locator (URL)

A web page, as a file, needs to have a unique identifier to distinguish it from other web pages.

URL is a standard for specifying any kind of information on the internet.URL defines four things: protocol, host computer, port and path.

Protocol. It is the client-server program that we need to access the web page.Example protocol is HTTP (HyperText Transfer Protocol) and FTP (File Transfer Protocol).

Host. The host identifier can be the IP address of the server or the unique name given to the server. IP addresses can be defined in dotted decimal notation.

Port. The port, a 16-bit integer, is normally predefined for the client-server application. For example, if the HTTP protocol is used for accessing the web page, the well-known port number is 80. However, if a different port is used, the number can be explicitly given.

Path. The path identifies the location and the name of the file in the underlying operating system.

The format of this identifier normally depends on the operating system. In UNIX, a path is a set of directory names followed by the file name, all separated by a slash.

For example, /top/next/last/myfile is a path that uniquely defines a file named my file, stored in the directory last, which itself is part of the directory next, which itself is under the directory top.

To combine these four pieces together, the uniform resource locator (URL) is used. It uses three different separators between the four pieces.

Web Documents

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active.

Static Documents

Static documents are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. The contents in the server can be changed, but the user cannot change them.When a client accesses the document, a copy of the document is sent. The user can then use a browser to see the document.Static documents are prepared using one of several languages: Hyper Text Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extensible Hypertext Markup Language (XHTML).

Dynamic Documents

A dynamic document is created by a web server whenever a browser requests the document.

When a request arrives, the web server runs an application program or a script thatcreates the dynamic document.The server returns the result of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another.

A simple example of a dynamic document is the retrieval of the time and date from a server.

Time and date are kinds of information that are dynamic in that they change from moment to moment. The client can ask the server to run a program such as the date program in UNIX and send the result of the program to the client.

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called active documents.For example, if we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place.

When a browser requests an active document, the server sends a copy of the document or a script.

The document is then run at the client (browser) site. One way to create an active document is to use Java applets, a program written in Java on the server. It is compiled and ready to be run. The document is in byte code (binary) format.