UNIT 3 APPLICATION LAYER

Structure **Page No.**

3.0 INTRODUCTION

The application layer contains a variety of protocols that are commonly needed by users. One widely-used application protocol is HTTP (Hyper Text Transfer Protocol), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news. The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services. Application-layer functions typically include identifying communication partners, determining resource availability, and synchronizing communication. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application that has data to transmit. When determining resource availability, the application layer must decide whether sufficient network or the requested communication exists. This unit covers the details of all important functions and services of Application layer including different protocols available in Application layer.

3.1 OBJECTIVES

After going through this unit, you will be able to:

- define and know the functions of Application layer
- Understand the working of Domain Name Server
- Know the features and services of Telnet and FTP
- Understand the Network Management issues
- Understanding of the WWW Client/Server architecture
- Know the Email Architecture and Services

3.2 CLIENT SERVER ARCHITECTURE

Client/server architecture means one or more computers is/are acting as client one computer is behaving as serve which preferably of high configuration. In simple words client is defined as requester of services and server is defined as provider of services. Client and Servers are nothing but programs, which generally run on different machines/computers.

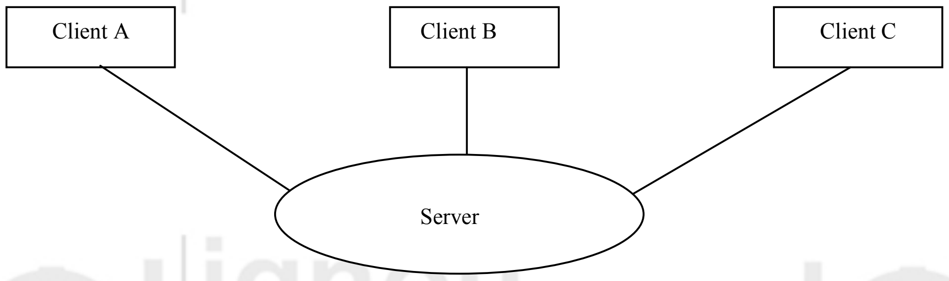


Figure 1: Client/Server Architecture

When client wants some information or wants to perform some task by server it has to contact the server as shown in figure1. Therefore, the client initially needs the address of the server, and tries to contact the server using that address. Server is a program which is always running and waiting to serve clients. Server welcomes the client (if free) as the client contacts it. After contacting the server, the server welcomes that client (if free), the client informs about its own address to servers so that server can reply the client also.

When both clients and servers exchange important information to each other connection establishes. Once with connection or link established, both client and server could exchange any information with each other. We will read more details of client/server architecture further in this unit during discussion about Word Wide Web.

3.3 DOMAIN NAME SERVER (DNS)

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 *ravi@128.111.45.46* means that if Ravi's ISP or organization moves the mail server to a different machine with a different IP address, his e-mail address has to change. Consequently, ASCII names were introduced to decouple machine names from machine addresses. In this way, Ravi's address might be something like ravi@art.ucsb.edu. Nevertheless, the network itself understands only numerical addresses, so some mechanism is required to convert the ASCII strings to network addresses.

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.

The essence of DNS is the invention of a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names and e-mail destinations to IP addresses but can also be used for other purposes. Very briefly, the way DNS is used as follows. To map a name onto an IP address, an application program calls a library procedure called the **resolver**, passing it the name as a parameter. The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to the resolver, which then returns it to the caller. Armed with the IP address, the program can then establish a TCP connection with the destination or send it UDP packets.

3.3.1 The DNS Name Space

Managing a large and constantly changing set of names is a nontrivial problem. In the postal system, name management is done by requiring letters to specify (implicitly or explicitly) the country, state or province, city, and street address of the addressee. Conceptually, the Internet is divided into over 200 top-level **domains**, where each domain covers many hosts. Each domain is partitioned into sub domains, and these are further partitioned, and so on. All these domains can be represented by a tree, as shown in Figure 2. The leaves of the tree represent domains that have no sub domains (but do contain machines, of course). A leaf domain may contain a single host, or it may represent a company and contain thousands of hosts.

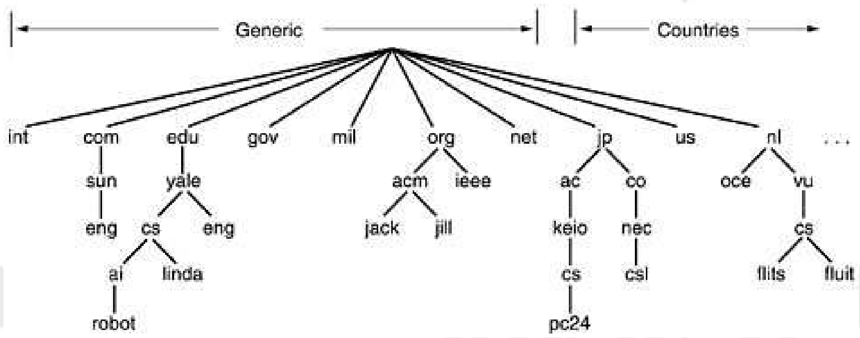


Figure 2: A portion of the Internet domain name space

The top-level domains come in two flavors: **generic** and **countries**. The original generic domains were com (commercial), edu (educational institutions), gov (the Government), int (certain international organizations), mil (the U.S. armed forces), net (network providers), and org (nonprofit organizations). The country domains include one entry for every country, as defined in ISO 3166.

In November 2000, ICANN approved four new, general-purpose, top-level domains, namely, biz (businesses), info (information), name (people's names), and pro (professions, such as doctors and lawyers). In addition, three more specialized top-level domains were introduced at the request of certain industries. These are aero (aerospace industry), coop (co-operatives), and museum (museums). Other top-level domains will be added in the future.

As an aside, as the Internet becomes more commercial, it also becomes more contentious. Take pro, for example. It was intended for certified professionals.

In general, getting a second-level domain, such as name-of-company.com, is easy. It merely requires going to a registrar for the corresponding top-level domain (com in this case) to check if the desired name is available and not somebody else's trademark. If there are no problems, the requester pays a small annual fee and gets the name. By now, virtually every common (English) word has been taken in the com domain. Try household articles, animals, plants, body parts, etc. Nearly all are taken.

Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced "dot"). Domain names can be either absolute or relative. An absolute domain name always ends with a period (e.g., eng.sun.com.), whereas a relative one does not. Relative names have to be interpreted in some context to uniquely determine their true meaning. In both cases, a named domain refers to a specific node in the tree and all the nodes under it.

Domain names are case insensitive, so edu, Edu, and EDU mean the same thing. Component names can be up to 63 characters long, and full path names must not exceed 255 characters.

In principle, domains can be inserted into the tree in two different ways. For example, cs.yale.edu could equally well be listed under the US country domain as cs.yale.ct.us. In practice, however, most organizations in the United States are under a generic domain, and most outside the United States are under the domain of their country. There is no rule against registering under two top-level domains, but few organizations except multinationals do it (e.g., sony.com and sony.nl).

Each domain controls how it allocates the domains under it. For example, Japan has domains ac.jp and co.jp that mirror edu and com. The Netherlands does not make this distinction and puts all organizations directly under nl. Thus, all three of the following are university computer science departments:

1. cs.yale.edu(Yale University, in the United States)
2. cs.vu.nl(Vrije Universiteit, in The Netherlands)
3. cs.keio.ac.jp(Keio University, in Japan)

To create a new domain, permission is required of the domain in which it will be included. For example, if a VLSI group is started at Yale and wants to be known as vlsi.cs.yale.edu, it has to get permission from whoever manages cs.yale.edu.

Similarly, if a new university is chartered, say, the University of Northern South Dakota, it must ask the manager of the edu domain to assign it unsd.edu. In this way, name conflicts are avoided and each domain can keep track of all its subdomains. Once a new domain has been created and registered, it can create subdomains, such as cs.unsd.edu, without getting permission from anybody higher up the tree.

Naming follows organizational boundaries, not physical networks. For example, if the computer science and electrical engineering departments are located in the same building and share the same LAN, they can nevertheless have distinct domains.

3.3.2 Resource Records

Every domain, whether it is a single host or a top-level domain, can have a set of **resource records** associated with it. For a single host, the most common resource record is just its IP address, but many other kinds of resource records also exist. When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus, the primary function of DNS is to map domain names onto resource records.

A resource record is a *five-tuple*. Although they are encoded in binary for efficiency, in most expositions, resource records are presented as ASCII text, one line per resource record. The format we will use is as follows:

- **Domain_name, Time_to_live, Class Type, Value**

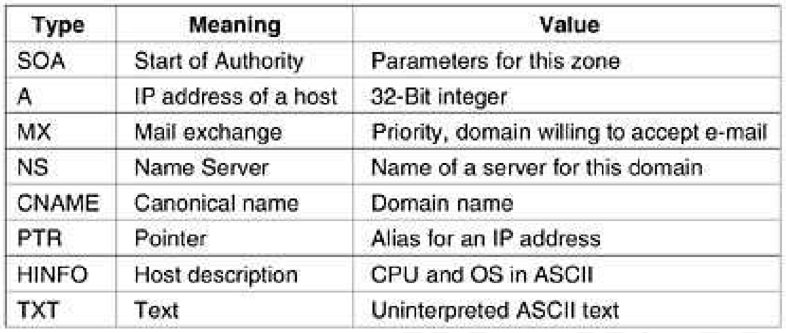
The Domain_name tells the domain to which this record applies. Normally, many records exist for each domain and each copy of the database holds information about multiple domains. This field is thus the primary search key used to satisfy queries. The order of the records in the database is not significant.

The Time_to_live field gives an indication of how stable the record is. Information that is highly stable is assigned a large value, such as 86400 (the number of seconds in 1 day). Information that is highly volatile is assigned a small value, such as 60 (1 minute).

The third field of every resource record is the Class. For Internet information, it is always IN. For non-Internet information, other codes can be used, but in practice, these are rarely seen.

The Type field tells what kind of record this is. The most important types are listed in Table. 1.2.2.

Table 1: The principal DNS Resource Record



Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

3.3.3 Name Servers

In theory at least, a single name server could contain the entire DNS database and respond to all queries about it. In practice, this server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled.

To avoid the problems associated with having only a single source of information, the DNS name space is divided into non-overlapping **zones**. One possible way to divide the name space is shown in figure 3. Each zone contains some part of the tree and also contains name servers holding the information about that zone.

Normally, a zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their information from the primary name server. To improve reliability, some servers for a zone can be located outside the zone.

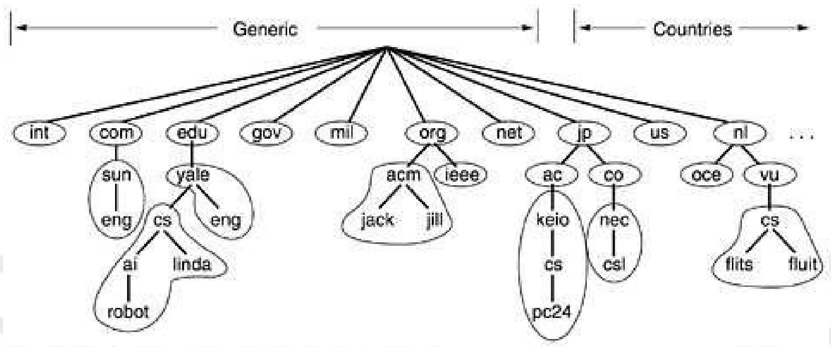


Figure 3: Part of the DNS name space showing the division into zones.

Where the zone boundaries are placed within a zone is up to that zone's administrator. This decision is made in large part based on how many name servers are desired, and where.

☞ Check Your Progress 1

1. Select the right choice.

   a) Which Layer is not present in TCP/IP model?
      (A) Application Layer (B) Internet Layer
      (C) Transport Layer (D) Presentation Layer

   …………………………………………………………………………………………
   …………………………………………………………………………………………

   b) Let most segment of a name inn DNS represents
      (A) Individual Network. (B) Individual computer.
      (C) Domain name (D) Network type.

   …………………………………………………………………………………………
   …………………………………………………………………………………………

   c) Address 192.5.48.3 belongs to
      (A) Class A. (B) Class B.
      (C) Class C. (D) Class D.

   …………………………………………………………………………………………
   …………………………………………………………………………………………

2. Discuss the DNS Name Space?

   …………………………………………………………………………………………
   …………………………………………………………………………………………

3. Write a note on Name Server?

……………………………………………………………………………………………

……………………………………………………………………………………………

3.4 REMOTE LOGIN (TELNET)

Telnet permits a user to connect to an account on a remote machine. A client program running on the user's machine communicates using the Telnet protocol with a server program running on the remote machine.

3.4.1 The Telnet Application

The user (say Ravi) has an account on both the local and remote machines.
For example, 'Ravi' on farkas.mcmaster.ca types telnet optlab.cas.mcmaster.ca at his user prompt. Here, farkas.mcmaster.ca is the client and optlab.cas.mcmaster.ca is the server. The Telnet client would perform a methotrexate() call to determine the IP address of optlab.cas.mcmaster.ca. Then the client would create a socket to communicate with the Telnet server. The server prompts 'Ravi' for a login identifier - the name of the user's account on the remote server followed by a password. The Telnet users interact with the remote machine in the same way they would interact with their local machine. The client relays Ravi's keystrokes to the remote server, and the remote server displays them on its pseudo terminal (which is actually the display screen on the client machine).

3.4.2 The Telnet Protocol

The Telnet client program performs two important functions **- interacting** with the user terminal on the local host and **exchanging messages** with the Telnet server. The client connects to port 23 on the remote machine, which is the port number reserved for Telnet servers. The TCP connection persists for the duration of the login session. The client and the server maintain the connection, even when the user interrupts the transfer of data, for example by hitting *cntl-C*. Since Telnet is designed to work over two hosts on different platforms, the protocol assumes that the two hosts run a Network Virtual Terminal (NVT).The TCP connection is set up across these two NVT terminals. The NVT is a very simple character device with a keyboard and a printer - data typed by the user on the keyboard is translated by the client software into NVT format and sent via its NVT terminal to the server, and data received in NVT format from the server is translated by the client into the local machine format and output to the printer.

The NVT terminals on the two hosts exchange data in the 7-bit U.S. variant of the ASCII format, with each character sent as an octet with the first bit set to19-21-10. Some control information, such as end-of-line indication, is transmitted as the character sequence CR (carriage return) followed by an LF (linefeed). Each Telnet control message starts with the special octet (Interpret as Command (IAC)) octet of all 1s) to ensure that the recipient interprets the subsequent octets as a command. Otherwise, each octet is interpreted as data (e.g., a user keystroke). Sending control messages on the same connection as the data is referred to as *inband signaling*. The initial control messages between the client and the server are used to exchange information about their capabilities (Telnet option negotiation). For example, the client may indicate the type and speed of its terminal, and whether data is to be sent one character or one line at a time. After the capabilities exchange, the server instructs the client to send a login identifier and password. Once the authentication completes, the user interacts directly with the remote machine. The client application relays user keystrokes to the server, and the server relays the output back to the client, using inband signaling, with the interpretation that commands follow the IAC octet of all ones. Telnet cannot rely on the conventional data stream alone to carry such control sequences between client and server.

3.5 FILE TRANSFER PROTOCAL (FTP)

FTP allows a user to copy files to and from a remote machine. The client program also sends commands to the server program to coordinate the copying of files between the two machines on behalf of the user.

3.5.1 The FTP Application

The FTP client connects to the remote machine which prompts the user to enter a login identifier and a password. However, some users may not have their own accounts on the remote machine. To grant access to a broad set of users, many FTP servers have a special account (e.g. *anonymous*) that does not require password information. Instead, the user logs in using *guest* or his email address as password. The FTP server coordinates access to a collection of files in various directories. In case of anonymous FTP, the server typically has a special directory, with one or more subdirectories, that can be accessed by the client. The user logged into the FTP server can traverse through the directories of files on the remote machine, and send or receive files. This is typically done via the command-line interface. The interface may also allow the client to send or receive multiple files with a single command. Recent FTP client applications provide a menu-based graphical user interface. For instance, a Web browser allows users to specify the desired file as an URL (e.g.,ftp://ftp.optlab.cas.mcmaster.ca/midterm.pdf). In this case, the web browser connects to the FTP server as an anonymous user and sends a sequence of FTP commands to fetch the requested file.

3.5.2 The FTP protocol

FTP differs from other applications such as Telnet since it uses separate TCP connections for control and data. Recall that in Telnet both control information and data are sent over the same TCP connection using in-band signaling. The two TCP connections in FTP are:

1. **The control connection** is established in the normal client-server fashion. In this case, the server does a *passive open* (is listening) on port 21 for FTP, and waits for the client connection. The client does an *active open* (the2nd handshake in a TCP connection) to establish the control connection. The client uses an ephemeral port number (above 1023) for the control connection. This control connection stays up for the entire time that the client communicates with this server. This connection is used for commands from the client to the server and for the server's replies. The IP type of service for the control connection should be to minimize delay in passing these commands over the TCP connection.

2. **A data connection** is created each time a file is transferred between the client and the server. The IP type of service for the data connection should be to maximize throughput since this connection is file transfer, and we want to send this entire file over a high bandwidth line. The specification of FTP includes more than 30 different commands, which are transmitted over the control connection in NVT ASCII format. The commands are not case-sensitive and may have arguments; each command ends with a two character sequence of a carriage return (CR) followed by a line feed (LF). It must be emphasized here that these commands are different from the commands typed by the user at the interface provided by the client. Transferring a single file for instance requires only a single user-level command (e.g., *put* or *get*), but this single command triggers the client to send a set of FTP commands to the server. The FTP server responds to each command with a three-digit reply code (for the FTP client) and an optional text message (for the user).

The control connection persists over a sequence of FTP commands, as the client and the server continue their dialogue. The typical interaction starts with a command that identifies the user on the server machine followed by another command to send the user password. The arguments for these commands are gleaned from the user's input (his account name and password). The server uses this information to verify whether the user has an authorized account on the remote machine, and in the case of anonymous FTP decides on the set of19-21-3directories to which the anonymous guest has access. The next set of commands depends on the user request to send, receive, or view the files in a present directory.

The actual file (data) transfer uses a separate TCP connection established by the host sending the data. For instance, if the user wants to retrieve the file *midterm.pdf* from the remote server, the server initiates the creation of the TCP data connection. In case, the user wants to put a file into the remote machine, it is the client who initiates the creation of the TCP connection. The data connection is usually established on port 20 on the server machine. In the former case (when the file is to be retrieved from the server), the server does not know the destination port for the FTP client. So before sending the command to retrieve the file, the client instructs its operating system to allocate a port number (above 1023) for such a transaction. This information is given to the server via the control connection. The data connection is created (using the usual TCP 3 way handshake), and the server writes the contents of the file, and closes the connection. The client reads the bytes from its socket upto the end of file (EOF) character. Also, unlike Telnet, FTP does not require the data transfer to 7 bit ASCII characters (NVT format); it actually permits a wide range of data types including binary files. The client requests the form of data transfer using the control connection. In practice, each data transfer requires a separate TCP connection. In contrast, the control connection can persist across multiple data transfers.

3.6 NETWORK MANAGEMENT

We can define **network management** as monitoring, testing, configuring, and troubleshooting network components to meet a set of requirements defined by an organization. These requirements include the smooth, efficient operation of the network that provides the predefined quality of service for users. To accomplish this task, a network management system uses hardware, software, and humans. In this chapter, first we briefly discuss the functions of a network management system. Then we concentrate on the most common management system, the Simple Network Management Protocol (SNMP).

We can say that the functions performed by a network management system can be divided into **five broad categories**: configuration management, fault management, performance management, security management, and accounting management.

3.6.1 Configuration Management

A large network is usually made up of hundreds of entities that are physically or logically connected to one another. These entities have an initial configuration when the network is set up, but can change with time. Desktop computers may be replaced by others; application software may be updated to a newer version; and users may move from one group to another. The **configuration management** system must know, at any time, the status of each entity and its relation to other entities. Configuration management can be divided into two subsystems: reconfiguration and documentation.

3.6.2 Reconfiguration

Reconfiguration, which means adjusting the network components and features, can be a daily occurrence in a large network. There are three types of reconfiguration: hardware reconfiguration, software reconfiguration, and user-account reconfiguration. Hardware reconfiguration covers all changes to the hardware. For example, a desktop computer may need to be replaced. A router may need to be moved to another part of the network. A sub network may be added or removed from the network. All these need the time and attention of network management. In a large network, there must be specialized personnel trained for quick and efficient hardware reconfiguration. Unfortunately, this type of reconfiguration cannot be automated and must be manually handled case by case.

Software reconfiguration covers all changes to the software. For example, new software may need to be installed on servers or clients. An operating system may need updating. Fortunately, most software reconfiguration can be automated. For example, updating an application on some or all clients can be electronically downloaded from the server.

User-account reconfiguration is not simply adding or deleting users on a system. You must also consider the user privileges, both as an individual and as a member of a group. For example, a user may have read and write permission with regard to some files, but only read permission with regard to other files. User-account reconfiguration can be, to some extent, automated. For example, in a college or university, at the beginning of each quarter or semester, new students are added to the system. The students are normally grouped according to the courses they take or the majors they pursue.

3.6.3 Documentation

The original network configuration and each subsequent change must be recorded meticulously. This means that there must be documentation for hardware, software, and user accounts. **Hardware documentation** normally involves two sets of documents: maps and specifications. Maps track each piece of hardware and its connection to the network. There can be one general map that shows the logical relationship between each sub-network. There can also be a second general map that shows the physical location of each sub-network. For each sub network, then, there are one or more maps that show all pieces of equipment. The maps use some kind of standardization to be easily read and understood by current and future personnel. Each piece of hardware also needs to be documented. There must be a set of specifications for each piece of hardware connected to the network. These specifications must include information such as hardware type, serial number, vendor (address and phone number), time of purchase, and warranty information. All software must also be documented. Software documentation includes information such as the software type, the version, the time installed, and the license agreement. Most operating systems have a utility that allows the documentation of user accounts and their privileges. The management must make sure that the files with this information are updated and secured. Some operating systems record access privileges in two documents one shows all files and access types for each user; the other shows the list of users that have a particular access to a file.

3.6.4 Fault Management

Complex networks today are made up of hundreds and sometimes thousands of components. Proper operation of the network depends on the proper operation of each component individually and in relation to each other. **Fault management** is the area of network management that handles this issue. An effective fault management system has two subsystems: reactive fault management and proactive fault management.

3.6.5 Reactive Fault Management

A reactive fault management system is responsible for detecting, isolating, correcting, and recording faults. It handles short-term solutions to faults. The first step taken by a reactive fault management system is to detect the exact location of the fault. A fault is defined as an abnormal condition in the system. When a fault occurs, either the system stops working properly or the system creates excessive errors. A good example of a fault is a damaged communication medium. This fault may interrupt communication or produce excessive errors.

The next step taken by a reactive fault management system is to isolate the fault. A fault, if isolated, usually affects only a few users. After isolation, the affected users are immediately notified and given an estimated time of correction. The third step is to correct the fault. This may involve replacing or repairing the faulty component(s).After the fault is corrected, it must be documented. The record should show the exact location of the fault, the possible cause, the action or actions taken to correct the fault, the cost, and time it took for each step. Documentation is extremely important for several reasons:

- The problem may recur. Documentation can help the present or future administrator or technician solve a similar problem.
- The frequency of the same kind of failure is an indication of a major problem in the system. If a fault happens frequently in one component, it should be replaced with a similar one, or the whole system should be changed to avoid the use of that type of component.
- The statistic is helpful to another part of network management, performance management.

3.6.6 Proactive Fault Management

Proactive fault management tries to prevent faults from occurring. Although this is not always possible, some types of failures can be predicted and prevented. For example, if a manufacturer specifies a lifetime for a component or a part of a component, it is a good strategy to replace it before that time. As another example, if a fault happens frequently at one particular point of a network, it is wise to carefully reconfigure the network to prevent the fault from happening again.

3.6.7 Performance Management

Performance management, which is closely related to fault management, tries to monitor and control the network to ensure that it is running as efficiently as possible. Performance management tries to quantify performance by using some measurable quantity such as capacity, traffic, throughput, or response time.

Capacity

One factor that must be monitored by a performance management system is the capacity of the network. Every network has a limited capacity, and the performance management system must ensure that it is not used above this capacity. For example, if a LAN is designed for 100 stations at an average data rate of2 Mbps, it will not operate properly if 200 stations are connected to the network. The data rate will decrease and blocking may occur.

Traffic

Traffic can be measured in two ways: internally and externally. Internal traffic is measured by the number of packets (or bytes) traveling inside the network for a given period. External traffic is measured by the exchange of packets (or bytes) outside the network. During peak hours, when the system is heavily used, blocking may occur if there is excessive traffic.

Throughput

We can measure the throughput of an individual device (such as a router) or a part of the network. Performance management monitors the throughput to make sure that it is not reduced to unacceptable levels.

Response Time

Response time is normally measured from the time a user requests a service to the time the service is granted. Other factors such as capacity and traffic can affect the response time. Performance management monitors the average response time and the peak-hour response time. Any increase in response time is a very serious condition as it is an indication that the network is working above its capacity.

3.6.8 Security Management

Security management is responsible for controlling access to the network based on the predefined policy.

3.6.9 Accounting Management

Accounting management is the control of users' access to network resources through charges. Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network. Charging does not necessarily mean cash transfer; it may mean debiting the departments or divisions for budgeting purposes. Today, organizations use an accounting management system for the following reasons:

- It prevents users from monopolizing limited network resources.
- It prevents users from using the system inefficiently.
- Network managers can do short- and long-term planning based on the demand for
- Network use.

3.6.10 SNMP Protocol

Since it was developed in 1988, the Simple Network Management Protocol has become the de facto standard for internetwork management. Since it is a simple solution, requiring little code to implement, vendors can easily build SNMP agents to their products. SNMP is extensible, allowing vendors to easily add network management functions to their existing products. SNMP also separates the management architecture from the architecture of the hardware devices, which broadens the base of multivendor support. Perhaps most important, unlike other so-called standards, SNMP is not a mere paper specification, but an implementation that is widely available today. To know more about SNMP you may refer to fifth semester course BCS-052 Network programming and Administration.

SNMP is based on the manager/agent model consisting of a manager, an agent, a database of management information, managed objects and the network protocol. The manager provides the interface between the human network manager and the management system. The agent provides the interface between the manager and the physical device(s) being managed (see the illustration above).

A typical agent usually

- Implements full SNMP protocol.
- Stores and retrieves management data as defined by the Management Information Base.
- Can asynchronously signal an event to the manager
- Can be a proxy for some non-SNMP manageable network node.

A typical manager usually

- Implemented as a Network Management Station (the NMS)
- Implements full SNMP Protocol Able to
- Query agents
- Get responses from agents
- Set's variables in agents
- Acknowledge's asynchronous events from agents.

☞ Check Your Progress 2

1. Select the right choice.

   a) FTP does not use

      (A) Two transfer mode.
      (B) Control connection to remote computer before file can be transferred.
      (C) User Datagram Protocol.
      (D) Authorization of a user through login and password verification.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   b) Protocol used to monitor and control network devices operates at:

      (A) Application layer (B) Transport layer
      (C) Network layer (D) Data Link layer

   ……………………………………………………………………………………

   ……………………………………………………………………………………

2. Discuss TCP connections in FTP.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

3. What is configuration management in computer network management?

   ……………………………………………………………………………………

   ……………………………………………………………………………………

3.7 WORD WIDE WEB AND CLIENT SERVER APPLICATIONS

The World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet. In the last decade on so 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.

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.

The initial proposal for a web of linked documents came from CERN physicist Tim Berners-Lee in March 1989. The first (text-based) prototype was operational 18 months later. In December 1991, a public demonstration was given at the Hypertext '91 conference in San Antonio, Texas.

This demonstration and its attendant publicity caught the attention of other researchers, which led Marc Andreessen at the University of Illinois to start developing the first graphical browser, Mosaic. It was released in February 1993. Mosaic was so popular that a year later, Andreessen left to form a company, Netscape Communications Corp., whose goal was to develop clients, servers, and other Web software. When Netscape went public in 1995, investors, apparently thinking this was the next Microsoft, paid $1.5 billion for the stock. This record was all the more surprising because the company had only one product, was operating deeply in the red, and had announced in its prospectus that it did not expect to make a profit for the foreseeable future. For the next three years, Netscape Navigator and Microsoft's Internet Explorer engaged in a "browser war," each one trying frantically to add more features (and thus more bugs) than the other one. In 1998, America Online bought Netscape Communications Corp. for $4.2 billion, thus ending Netscape's brief life as an independent company.

In 1994, CERN and M.I.T. signed an agreement setting up the **World Wide Web Consortium** (sometimes abbreviated as **W3C**), an organization devoted to further developing the Web, standardizing protocols, and encouraging interoperability between sites. Berners-Lee became the director. Since then, several hundred universities and companies have joined the consortium. Although there are now more books about the Web than you can shake a stick at, the best place to get up-to-date information about the Web is (naturally) on the Web itself.

3.7.1 Architectural Overview (WWW)

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. Web pages, starts with a title, contain 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 non graphical browsers, such as Lynx, exist, they are not as popular as graphical browsers. Voice-based browsers are also being developed.

The basic model of how the Web works is shown in Figure 4, Here the browser is displaying a Web page on the client machine. When the user clicks on a line of text that is linked to a page on the *abcd.com* server, the browser follows the hyperlink by sending a message to the *abcd.com* server asking it for the page. When the page arrives, it is displayed. If this page contains a hyperlink to a page on the *xyz.com* server that is clicked on, the browser then sends a request to that machine for the page, and so on indefinitely.

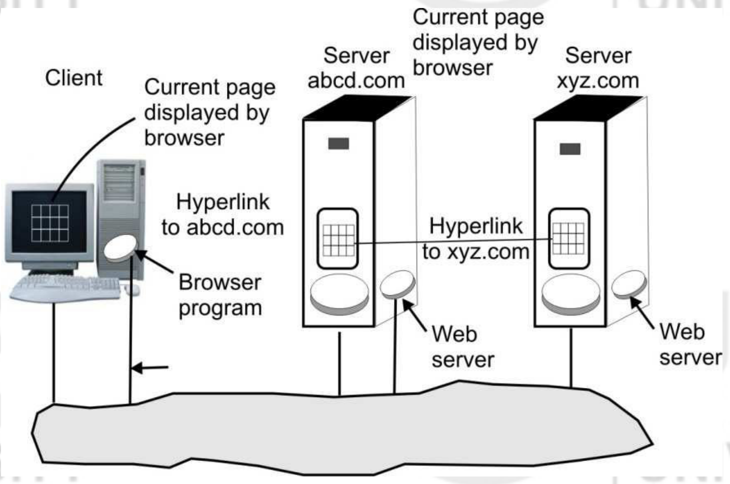


Figure 4: The parts of the Web model.

The Client Side

In essence, a browser is a program that can display a Web page and catch mouse clicks to items on the displayed page. When an item is selected, the browser follows the hyperlink and fetches the page selected. Therefore, the embedded hyperlink needs a way to name any other page on the Web. Pages are named using **URLs** (**Uniform Resource Locators**). A typical URL is http://www.abcd.com/products.html

It is sufficient to know that a URL has three parts: the name of the protocol (*http*), the DNS name of the machine where the page is located (*www.abcd.com*), and (usually) the name of the file containing the page (*products.html*).

When a user clicks on a hyperlink, the browser carries out a series of steps in order to fetch the page pointed to. Suppose that a user is browsing the Web and finds a link on Internet telephony that point to ITU's home page, which is *http://www.itu.org/home/index.html*. Let us trace the steps that occur when this link is selected.

1. The browser determines the URL (by seeing what was selected).

2. The browser asks DNS for the IP address of www.itu.org.
3. DNS replies with 156.106.192.32.
4. The browser makes a TCP connection to port 80 on 156.106.192.32.
5. It then sends over a request asking for file /home/index.html.
6. The www.itu.org server sends the file /home/index.html.
7. The browser displays all the text in /home/index.html.
8. The browser fetches and displays all images in this file.

Many browsers display which step they are currently executing in a status line at the bottom of the screen. In this way, when the performance is poor, the user can see if it is due to DNS not responding, the server not responding, or simply network congestion during page transmission.

Although a browser is basically an HTML interpreter, most browsers have numerous buttons and features to make it easier to navigate the Web. Most have a button for going back to the previous page, a button for going forward to the next page (only operative after the user has gone back from it), and a button for going straight to the user's own start page. Most browsers have a button or menu item to set a bookmark on a given page and another one to display the list of bookmarks, making it possible to revisit any of them with only a few mouse clicks. Pages can also be saved to disk or printed. Numerous options are generally available for controlling the screen layout and setting various user preferences.

In addition to having ordinary text (not underlined) and hypertext (underlined), Web pages can also contain icons, line drawings, maps, and photographs. Each of these can (optionally) be linked to another page. Clicking on one of these elements causes the browser to fetch the linked page and display it on the screen, the same as clicking on text. With images such as photos and maps, which page is fetched next may depend on what part of the image was clicked on.

Not all pages contain HTML. A page may consist of a formatted document in PDF format, an icon in GIF format, a photograph in JPEG format, a song in MP3 format, a video in MPEG format, or any one of hundreds of other file types. Since standard HTML pages may link to any of these, the browser has a problem when it encounters a page it cannot interpret.

Rather than making the browsers larger and larger by building in interpreters for a rapidly growing collection of file types, most browsers have chosen a more general solution. When a server returns a page, it also returns some additional information about the page. This information includes the MIME type of the page. Pages of type *text/html* are just displayed directly, as are pages in a few other built-in types. If the MIME type is not one of the built-in ones, the browser consults its table of MIME types to tell it how to display the page.

There are two possibilities: plug-ins and helper applications. A **plug-in** is a code module that the browser fetches from a special directory on the disk and installs as an extension to itself, as illustrated in Figure 5(a) Because plug-ins run inside the browser, they have access to the current page and can modify its appearance. After the plug-in has done its job (usually after the user has moved to a different Web page), the plug-in is removed from the browser's memory.

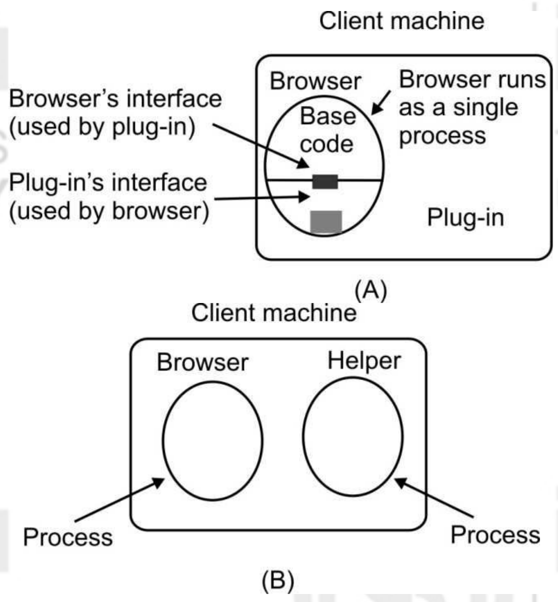

Client machine
Browser
Browser runs as a single process
Browser's interface (used by plug-in)
Base code
Plug-in's interface (used by browser)
Plug-in
(A)
Client machine
Browser
Helper
Process
Process
(B)

Figure 5: (a) A browser plug-in. (b) A helper application.

Each browser has a set of procedures that all plug-ins must implement so the browser can call the plug-in. For example, there is typically a procedure the browser's base code calls to supply the plug-in with data to display. This set of procedures is the plug-in's interface and is browser specific.

In addition, the browser makes a set of its own procedures available to the plug-in, to provide services to plug-ins. Typical procedures in the browser interface are for allocating and freeing memory, displaying a message on the browser's status line, and querying the browser about parameters.

Before a plug-in can be used, it must be installed. The usual installation procedure is for the user to go to the plug-in's Web site and download an installation file. On Windows, this is typically a self-extracting zip file with extension *.exe*. When the zip file is double clicked, a little program attached to the front of the zip file is executed. This program unzips the plug-in and copies it to the browser's plug-in directory. Then it makes the appropriate calls to register the plug-in's MIME type and to associate the plug-in with it. On UNIX, the installer is often a shell script that handles the copying and registration.

The other way to extend a browser is to use a **helper application**. This is a complete program, running as a separate process. It is illustrated in Figure 5(b). Since the helper is a separate program, it offers no interface to the browser and makes no use of browser services. Instead, it usually just accepts the name of a scratch file where the content file has been stored, opens the file, and displays the contents. Typically, helpers are large programs that exist independently of the browser, such as Adobe's Acrobat Reader for displaying PDF files or Microsoft Word. Some programs (such as Acrobat) have a plug-in that invokes the helper itself.

Many helper applications use the MIME type *application*. A considerable number of subtypes have been defined, for example, *application/pdf* for PDF files and *application/msword* for Word files. In this way, a URL can point directly to a PDF or Word file, and when the user clicks on it, Acrobat or Word is automatically started and handed the name of a scratch file containing the content to be displayed. Consequently, browsers can be configured to handle a virtually unlimited number of document types with no changes to the browser. Modern Web servers are often configured with hundreds of type/subtype combinations and new ones are often added every time a new program is installed.

Helper applications are not restricted to using the *application* MIME type. Adobe Photoshop uses *image/x-photoshop* and Real One Player is capable of handling *audio/mp3*, for example.

On Windows, when a program is installed on the computer, it registers the MIME types it wants to handle. This mechanism leads to conflict when multiple viewers are available for some subtype, such as *video/mpg*. What happens is that the last program to register overwrites existing (MIME type, helper application) associations, capturing the type for itself. As a consequence, installing a new program may change the way a browser handles existing types.

On UNIX, this registration process is generally not automatic. The user must manually update certain configuration files. This approach leads to more work but fewer surprises.

Browsers can also open local files, rather than fetching them from remote Web servers. Since local files do not have MIME types, the browser needs some way to determine which plug-in or helper to use for types other than its built-in types such as *text/html* and *image/jpeg*. To handle local files, helpers can be associated with a file extension as well as with a MIME type. With the standard configuration, opening *foo.pdf* will open it in Acrobat and opening *bar.doc* will open it in Word. Some browsers use the MIME type, the file extension, and even information taken from the file itself to guess the MIME type. In particular, Internet Explorer relies more heavily on the file extension than on the MIME type when it can.

Here, too, conflicts can arise since many programs are willing, in fact, eager, to handle, say, *.mpg*. During installation, programs intended for professionals often display checkboxes for the MIME types and extensions they are prepared to handle to allow the user to select the appropriate ones and thus not overwrite existing associations by accident. Programs aimed at the consumer market assume that the user does not have a clue what a MIME type is and simply grab everything they can without regard to what previously installed programs have done.

The ability to extend the browser with a large number of new types is convenient but can also lead to trouble. When Internet Explorer fetches a file with extension *exe*, it realizes that this file is an executable program and therefore has no helper. The obvious action is to run the program. However, this could be an enormous security hole. All a malicious Web site has to do is produce a Web page with pictures of, say, movie stars or sports heroes, all of which are linked to a virus. A single click on a picture then causes an unknown and potentially hostile executable program to be fetched and run on the user's machine. To prevent unwanted guests like this, Internet Explorer can be configured to be selective about running unknown programs automatically, but not all users understand how to manage the configuration.

The Server Side

Now let us take a look at the server side. As we saw above, when the user types in a URL or clicks on a line of hypertext, the browser parses the URL and interprets the part between *http://* and the next slash as a DNS name to look up. Armed with the IP address of the server, the browser establishes a TCP connection to port 80 on that server. Then it sends over a command containing the rest of the URL, which is the name of a file on that server. The server then returns the file for the browser to display.

The steps that the server performs in its main loop are as follows:

1. Accept a TCP connection from a client (a browser).
2. Get the name of the file requested.
3. Get the file (from disk).
4. Return the file to the client.
5. Release the TCP connection.

Modern Web servers have more features, but in essence, this is what a Web server does.

A problem with this design is that every request requires making a disk access to get the file. The result is that the Web server cannot serve more requests per second than it can make disk accesses.

One obvious improvement (used by all Web servers) is to maintain a cache in memory of the *n* most recently used files. Before going to disk to get a file, the server checks the cache. If the file is there, it can be served directly from memory, thus eliminating the disk access. Although effective caching requires a large amount of main memory and some extra processing time to check the cache and manage its contents, the savings in time are nearly always worth the overhead and expense.

The next step for building a faster server is to make the server multithreaded. In one design, the server consists of a front-end module that accepts all incoming requests and *k* processing modules, as shown in Figure 6. The *k* + 1 threads all belong to the same process so the processing modules all have access to the cache within the process' address space. When a request comes in, the front end accepts it and builds a short record describing it. It then hands the record to one of the processing modules. In another possible design, the front end is eliminated and each processing module tries to acquire its own requests, but a locking protocol is then required to prevent conflicts.

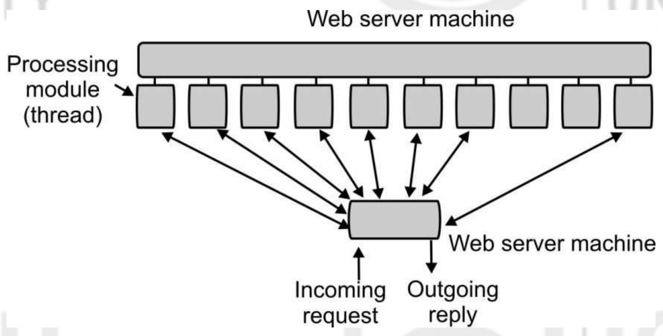


Figure 6: A multithreaded Web server with a front end and processing modules.

The processing module first checks the cache to see if the file needed is there. If so, it updates the record to include a pointer to the file in the record. If it is not there, the processing module starts a disk operation to read it into the cache (possibly discarding some other cached files to make room for it). When the file comes in from the disk, it is put in the cache and also sent back to the client.

The advantage of this scheme is that while one or more processing modules are blocked waiting for a disk operation to complete (and thus consuming no CPU time), other modules can be actively working on other requests. Of course, to get any real improvement over the single-threaded model, it is necessary to have multiple disks, so more than one disk can be busy at the same time. With *k* processing modules and *k* disks, the throughput can be as much as *k* times higher than with a single-threaded server and one disk.

In theory, a single-threaded server and *k* disks could also gain a factor of *k*, but the code and administration are far more complicated since normal blocking READ system calls cannot be used to access the disk. With a multithreaded server, they can be used since then a READ blocks only the thread that made the call, not the entire process.

Modern Web servers do more than just accept file names and return files. In fact, the actual processing of each request can get quite complicated. For this reason, in many servers each processing module performs a series of steps. The front end passes each incoming request to the first available module, which then carries it out using some subset of the following steps, depending on which ones are needed for that particular request.

1. Resolve the name of the Web page requested.
2. Authenticate the client.
3. Perform access control on the client.
4. Perform access control on the Web page.
5. Check the cache.
6. Fetch the requested page from disk.
7. Determine the MIME type to include in the response.
8. Take care of miscellaneous odds and ends.
9. Return the reply to the client.
10. Make an entry in the server log.

Step 1 is needed because the incoming request may not contain the actual name of the file as a literal string. For example, consider the URL *http://www.cs.vu.nl*, which has an empty file name. It has to be expanded to some default file name. Also, modern browsers can specify the user's default language (e.g., Italian or English), which makes it possible for the server to select a Web page in that language, if available. In general, name expansion is not quite so trivial as it might at first appear, due to a variety of conventions about file naming.

Step 2 consists of verifying the client's identity. This step is needed for pages that are not available to the general public. We will discuss one way of doing this later in this chapter.

Step 3 checks to see if there are restrictions on whether the request may be satisfied given the client's identity and location. Step 4 checks to see if there are any access restrictions associated with the page itself. If a certain file (e.g., *.htaccess*) is present in the directory where the desired page is located, it may restrict access to the file to particular domains, for example, only users from inside the company.
Steps 5 and 6 involve getting the page. Step 6 needs to be able to handle multiple disk reads at the same time.

Step 7 is about determining the MIME type from the file extension, first few words of the file, a configuration file, and possibly other sources.

Step 8 is for a variety of miscellaneous tasks, such as building a user profile or gathering certain statistics.

Step 9 is where the result is sent back and step 10 makes an entry in the system log for administrative purposes. Such logs can later be mined for valuable information about user behavior, for example, the order in which people access the pages.

If too many requests come in each second, the CPU will not be able to handle the processing load, no matter how many disks are used in parallel. The solution is to add more nodes (computers), possibly with replicated disks to avoid having the disks become the next bottleneck. This leads to the **server farm** model of Figure 7. A front end still accepts incoming requests but sprays them over multiple CPUs rather than multiple threads to reduce the load on each computer. The individual machines may themselves be multithreaded and pipelined as before.

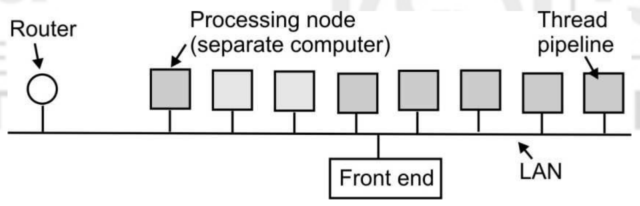


Figure 7: A server farm.

One problem with server farms is that there is no longer a shared cache because each processing node has its own memory unless an expensive shared-memory multiprocessor is used. One way to counter this performance loss is to have a front end keep track of where it sends each request and send subsequent requests for the same page to the same node. Doing this makes each node a specialist in certain pages so that cache space is not wasted by having every file in every cache. Another problem with server farms is that the client's TCP connection terminates at the front end, so the reply must go through the front end. This situation is depicted in Figure 8(a), where the incoming request (1) and outgoing reply (4) both pass through the front end.

Sometimes a trick, called **TCP handoff**, is used to get around this problem. With this trick, the TCP end point is passed to the processing node so it can reply directly to the client, shown in Figure 8(b). This handoff is done in a way that is transparent to the client.

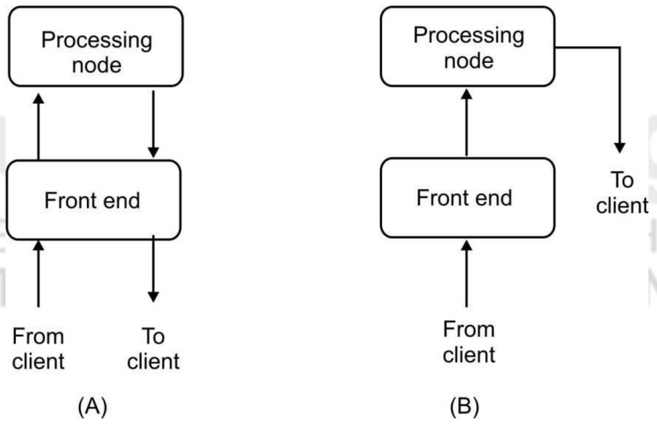


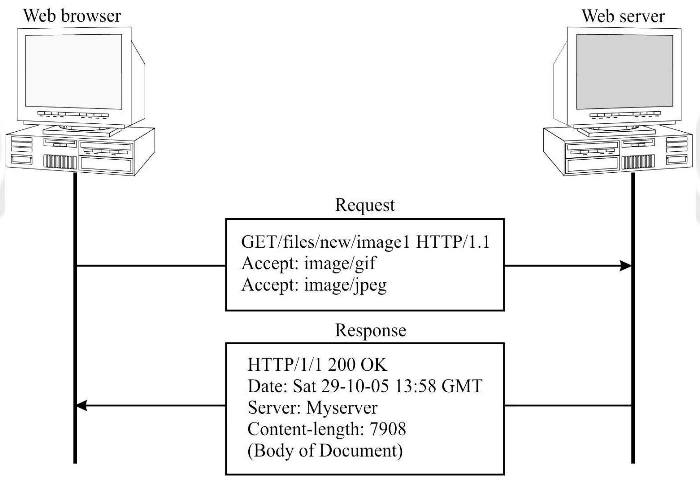
Figure 8: (a) Normal request-reply message sequence. (b) Sequence when TCP handoff is used.

HTTP

Hyper Text Transfer Protocol (HTTP) is used mainly to transfer data on World Wide Web. The commands from the client are embedded in a request message .The contents of the request message are embedded in a response message. HTTP uses the services of TCP at port 80.

HTTP is a stateless protocol since each transaction is independent of the previous transaction. The TCP connection between the client and the server is established for every page. It does not remember anything about the previous request. Keeping HTTP stateless was aimed at making the Web simple.

Sample HTTP request and response transaction is shown below:



```
GET/files/new/image1 HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1/1 200 OK
Date: Sat 29-10-05 13:58 GMT
Server: Myserver
Content-length: 7908
(Body of Document)
```

Figure 9: HTTP request and response

As shown in figure 9 above, the GET command requests the web server at www.myserver.com for an image file image1 .The HTTP/1.0 indicates that the browser uses the 1.0 version of the HTTP protocol.

The first line of response indicates that the server is also using HTTP version 1.0 .the return code of 200 indicates that the server processed the request successfully. Request message can be of following types:

Method	**Action**
GET	Requests a document from the server
HEAD	Request information about the document but not the document itself; i.e. head of the HTML page
POST	Sends some information from client to server. It appends the data to the existing document.
PUT	Sends a document from to server. It replaces the existing document.
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Enquire about available options

3.8 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.

The first e-mail systems simply consisted of file transfer protocols, with the convention that the first line of each message (i.e., file) contained the recipient's address. As time went on, the limitations of this approach became more obvious. Some of the complaints were as follows:

1. Sending a message to a group of people was inconvenient. Managers often need this facility to send memos to all their subordinates.

2. Messages had no internal structure, making computer processing difficult. For example, if a forwarded message was included in the body of another message, extracting the forwarded part from the received message was difficult.

3. The originator (sender) never knew if a message arrived or not.

4. If someone was planning to be away on business for several weeks and wanted all incoming e-mail to be handled by his secretary, this was not easy to arrange.

5. The user interface was poorly integrated with the transmission system requiring users first to edit a file, then leave the editor and invoke the file transfer program.

6. It was not possible to create and send messages containing a mixture of text, drawings, facsimile, and voice.

3.8.1 Architecture and Services

In this section we will provide an overview of what e-mail systems can do and how they are organized. They normally consist of two subsystems: the **user agents**, which allow people to read and send e-mail, and the **message transfer agents**, which move the messages from the source to the destination. The user agents are local programs that provide a command-based, menu-based, or graphical method for interacting with the e-mail system. The message transfer agents are typically system **daemons**, that is, processes that run in the background. Their job is to move e-mail through the system. Typically, e-mail systems support **five basic functions**.

Composition refers to the process of creating messages and answers. Although any text editor can be used for the body of the message, the system itself can provide assistance with addressing and the numerous header fields attached to each message. For example, when answering a message, the e-mail system can extract the originator's address from the incoming e-mail and automatically insert it into the proper place in the reply.

Transfer refers to moving messages from the originator to the recipient. In large part, this requires establishing a connection to the destination or some intermediate machine, outputting the message, and releasing the connection. The e-mail system should do this automatically, without bothering the user.

Reporting has to do with telling the originator what happened to the message. Was it delivered? Was it rejected? Was it lost? Numerous applications exist in which confirmation of delivery is important and may even have legal significance ("Well, Your Honor, my e-mail system is not very reliable, so I guess the electronic subpoena just got lost somewhere").

Displaying incoming messages is needed so people can read their e-mail. Sometimes conversion is required or a special viewer must be invoked, for example, if the message is a PostScript file or digitized voice. Simple conversions and formatting are sometimes attempted as well.

Disposition is the final step and concerns what the recipient does with the message after receiving it. Possibilities include throwing it away before reading, throwing it away after reading, saving it, and so on. It should also be possible to retrieve and reread saved messages, forward them, or process them in other ways.

In addition to these basic services, some e-mail systems, especially internal corporate ones, provide a variety of advanced features. Let us just briefly mention a few of these. When people move or when they are away for some period of time, they may want their e-mail forwarded, so the system should be able to do this automatically. Most systems allow users to create **mailboxes** to store incoming e-mail. Commands are needed to create and destroy mailboxes, inspect the contents of mailboxes, insert and delete messages from mailboxes, and so on.

Corporate managers often need to send a message to each of their subordinates, customers, or suppliers. This gives rise to the idea of a **mailing list**, which is a list of e-mail addresses. When a message is sent to the mailing list, identical copies are delivered to everyone on the list.

3.8.2 The User Agent

E-mail systems have two basic parts, as we have seen: **the user agents** and **the message transfer agents**. In this section, we will look at the user agents. A user agent is normally a program (sometimes called a mail reader) that accepts a variety of commands for composing, receiving, and replying to messages, as well as for manipulating mailboxes. Some user agents have a fancy menu- or icon-driven interface that requires a mouse, whereas others expect 1-character commands from the keyboard. Functionally, these are the same. Some systems are menu-or icon-driven but also have keyboard shortcuts.

Sending E-mail

To send an e-mail message, a user must provide the message, the destination address, and possibly some other parameters.

Reading E-mail

Typically, when a user agent is started up, it looks at the user's mailbox for incoming e-mail before displaying anything on the screen. Then it may announce the number of messages in the mailbox or display a one-line summary of each one and wait for a command.

SMTP server (Simple Mail Transfer Protocol)

For email messaging, every domain maintains an email server. The server runs protocols software that enable email communication. There are two main emails protocols: POP and SMTP. Because both the email protocol software programs run on server computers, the server computers are themselves called POP server and SMTP server. A single server can host both the POP and SMTP server programs.
SMTP is the Internet protocol used to transfer electronic mail between computers. The second generation of SMTP is called ESMTP (for Extended SMTP), but the differences are not important for this introduction.

It actually transfers the email message from the SMTP server of the sender to the SMTP server of the recipient. Its main job is to carry the message between the sender and the receiver. It uses TCP/IP underneath. That is, it runs on top of TCP/IP.
At the sender's site, an SMTP server takes the message sent by a user's computer. The SMTP server at the sender's end then transfers the message to the SMTP server of the recipient.

The SMTP server at the recipient's end takes the message and stores it in the appropriate user's mailbox.

☞ **Check Your Progress 3**

1. Select the right choice

    a) Internet's initial development was supported by:
    (A) ARPANET
    (B) Bill Rogers
    (C) Bill Gates
    (D) Microsoft

    …………………………………………………………………………………………

    …………………………………………………………………………………………

    b) What are the uses of the Internet?

    (A) Communication
    (B) Information Retrieval
    (C) Presentation of Information
    (D) All of the above

    …………………………………………………………………………………………

    …………………………………………………………………………………………

    c) net domain is used for

    (A) Educational institution
    (B) Internet infrastructure and service providers
    (C) International organizations
    (D) None of the above

    …………………………………………………………………………………………

    …………………………………………………………………………………………

2. Differentiate between http and ftp.

……………………………………………………………………………………

……………………………………………………………………………………

3.9 SUMMARY

This completes our discussion on the application layer. Include identifying communication partners, determining resource availability, and synchronizing communication. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application with data to transmit. When determining resource availability, the application layer must decide whether sufficient network or the requested communication exists. The unit very well defines the concept of DNS and various internet and communication related issues like www, emailing system, FTP, Telnet etc. To know further about different application layer protocol students may refer to the course material of BCS-052: Network Programming and Administration

3.10 REFERENCES/FURTHER READING

1. Introduction to Data Communication & Networking, 3rd Edition, Behrouz Forouzan, Tata McGraw Hill.
2. Computer Networks, A. S. Tanenbaum 4th Edition, Practice Hall of India, New Delhi. 2003.
3. Douglas E. Comer, Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture (4th Edition).
4. James F. Kurose, Computer Networking: A Top-Down Approach Featuring the Internet (3rd Edition).
5. Larry L. Peterson, Computer Networks: A Systems Approach, 3rd Edition (The Morgan Kaufmann Series in Networking).
6. www. wikipedia.org
7. W. Richard Stevens, The Protocols (TCP/IP Illustrated, Volume 1).
8. William Stallings, Data and Computer Communications, Seventh Edition.

3.11 SOLUTIONS / ANSWERS

☞ **Check Your Progress 1**

1. a) D

   b) B

   c) C

2. Managing a large and constantly changing set of names is a nontrivial problem. In the postal system, name management is done by requiring letters to specify (implicitly or explicitly) the country, state or province, city, and street address of the addressee. Conceptually, the Internet is divided into over 200 top-level **domains**, where each domain covers many hosts. Each domain is partitioned into sub domains, and these are further partitioned, and so on. All these domains can be represented by a tree. The top-level domains come in two flavors: **generic** and **countries**.

3. A single name server could contain the entire DNS database and respond to all queries about it. In practice, this server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled. To avoid the problems associated with having only a single source of information, the DNS name space is divided into non-overlapping **zones**. A zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their information from the primary name server. To improve reliability, some servers for a zone can be located outside the zone.

☞ Check Your Progress 2

1. a) (C)

   b) (A)

2. Two TCP connections in FTP are:

   i) **The control connection** is established in the normal client-server fashion. In this case, the server does a *passive open* (is listening) on port 21 for FTP, and waits for the client connection. The client does an *active open* (the2nd handshake in a TCP connection) to establish the control connection. The client uses an ephemeral port number (above 1023) for the control connection. This control connection stays up for the entire time that the client communicates with this server. This connection is used for commands from the client to the server and for the server's replies. The IP type of service for the control connection should be to minimize delay in passing these commands over the TCP connection.

   ii) **A data connection** is created each time a file is transferred between the client and the server. The IP type of service for the data connection should be to maximize throughput since this connection is file transfer, and we want to send this entire file over a high bandwidth line. The specification of FTP includes more than 30 different commands, which are transmitted over the control connection in NVT ASCII format. The commands are not case-sensitive and may have arguments; each command ends with a two character sequence of a carriage return (CR) followed by a line feed (LF). It must be emphasized here that these commands are different from the commands typed by the user at the interface provided by the client. Transferring a single file for instance requires only a single user-level command (e.g., *put* or *get*), but this single command triggers the client to send a set of FTP commands to the server. The FTP server responds to each command with a three-digit reply code (for the FTP client) and an optional text message (for the user).

3. **Configuration management** system must know, at any time, the status of each entity and its relation to other entities. Configuration management can be divided into two subsystems: reconfiguration and documentation.

   Reconfiguration, which means adjusting the network components and features, can be a daily occurrence in a large network. There are three types of reconfiguration: hardware reconfiguration, software reconfiguration, and user-account reconfiguration. Hardware reconfiguration covers all changes to the hardware. For example, a desktop computer may need to be replaced. A router may need to be moved to another part of the network. A sub network may be added or removed from the network.

   The original network configuration and each subsequent change must be recorded meticulously. This means that there must be documentation for hardware, software, and user accounts. **Hardware documentation** normally involves two sets of documents: maps and specifications. Maps track each piece of hardware and its connection to the network. There can be one general map that shows the logical relationship between each sub-network. There can also be a second general map that shows the physical location of each sub-network. For each sub network, then, there are one or more maps that show all pieces of equipment. The maps use some kind of standardization to be easily read and understood by current and future personnel. Each piece of hardware also needs to be documented. There must be a set of specifications for each piece of hardware connected to the network

☞ Check Your Progress 3

1. a) (A)
   b) (D)
   c) (C)

2. FTP and HTTP were developed to make Internet transmission better. FTP is used to exchange files between computer accounts, to transfer files between an account and a desktop computer (upload), or to access software archives on the Internet. It's also commonly used to download programs and other files to your computer from other servers. It transfers files in two different formats ASCII for text files and Binary format for binary files. This allows a user to perform basic file and directory management operations such as deleting, copying, or renaming. HTTP is used primarily in today's society as a set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. It also provides access to other protocols like FTP, SMTP, NNTP, WAIS, Gopher, Telnet, and TN3270. Essential concepts that are part of HTTP include (as its name implies) the idea that files can contain references to other files whose selection will elicit additional transfer requests. Any web server machine contains, in addition to the HTML and other files it can serve, an HTTP daemon, a program that is designed to wait for HTTP requests and handle them when they arrive.