UNIT 1 GRAPHICAL USER INTERFACE

Structure **Page Nos.**

1.0 INTRODUCTION

The **Graphical User Interface (GUI)** is one of the most revolutionary changes to occur in the evolution of the modern computing system. The interaction between man and computer in the design of operating system has changed from a character-oriented system to the now more **graphics-oriented** system. This revolution has increased the accessibility and usability of computer systems to the general public.

We will look at GUI features of the various operating systems with little knowledge of operating system or memorized commands. The software providing such features supports **Modern User Interface** concept such as desktop metaphor, which makes computers available to the majority of people who are either novices or non-programmers. The personal computer was invented for these users.

We will start with the basic definition move on to basic components and finally look into some systems to find out what GUI facilities they are supporting.

1.1 OBJECTIVES

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

- define what GUI is and how it is different from character oriented system,
- define all the terms related with GUI, and
- identify important features of several GUIs.

1.2 WHAT IS GRAPHICAL USER INTERFACE?

The term "user interface" originated in the engineering environment in the late 1970s. Virtually everyone who interacted directly with computers had been an engineer and a programmer, but new kinds of users were emerging: the non-programming user. These users often reacted more negatively to difficulties in dealing with a machine. New forms of interaction were needed, new interfaces, were required, attention flowed to "the user interface".

With the introduction of the Macintosh in 1984, Apple Computer popularised the user interface as it is known today. Apple's user interface is now commonly referred to as a Graphical User Interface or GUI. The GUI has become associated with a common feature set available in a number of product offerings. Common features include:

- Secondary user-input devices. Usually a pointing device and typically a **mouse**.
- Point and shoot functionality with **screen menus** that **appear** or **disappear** under pointing-device-control.
- **Icons** that represent files, directories and other application and system entities.
- **Dialog boxes, button, sliders, check boxes** and many other **graphical metaphors** that let the programmer and user tell the computer what to do and how to do it.

Today's GUIs have expanded basic functionalities to support not only graphics, but also dimensions, color, height, video, and highly dynamic interaction. Modern user interfaces can simulate a very realistic view of a real, three-dimensional world.

1.3 EVOLUTION OF HUMAN AND MACHINE INTERACTION

The primary means of communication with computers earlier had been through command-based interfaces. In command interfaces, users have to learn a large set of commands to get their job(s) done. In earlier computer systems paper tapes, cards and batch jobs were the primary means of communicating these commands to the computers. Later, time-sharing systems allowed the use of CRT terminals to interact/communicate with the computer. These early systems were heavily burdened by users trying to share precious computer resources such as CPU and peripherals.

The batch systems and time-sharing led to command-driven user interfaces. Users had to memorise commands and options or consult a large set of user manuals. The early mainframe and minicomputer systems required a large set of instruction manuals on how to use the system. In some systems, meaningful terms were used for command names to help the end-user. But in other systems the end-user had to memorise several sequences of keystrokes to accomplish certain tasks.

Early users of computes were engineers and what we now call expert users; users who had a lot of interest in knowing more about computer systems and the technology. Command line interfaces were acceptable to the majority of these users. In the 1970s, computers were introduced to a new class of users: secretaries, managers and non-technical people. These new users were less interested in learning computer technology and more interested in getting their jobs done through the machine. The command-based interfaces caused many of these new users to develop computer phobia. Imagine the thought of memorising commands made up of "Control-Alt-Del" to boot the system.

To make life easier for the end –user, a large collection of devices have been invented to control, monitor and display information. The early (and still widely used) peripherals are the keyboard and the video terminal. But, it was not until the late 70s, that research projects at some universities led to the invention of pointing devices and windowing systems. The mouse and joystick were among some of the few pointing devices that were invented in this period. Also, research pioneers invented the notion of splitting the screen to allow multiple windows and direct manipulation of objects.

In the 70s, researchers designed powerful new workstations armed with graphical user-interfaces. The basic assumption of these new workstations was that one user could have a powerful desktop computer totally dedicated to that user's task. Thus, the computer is not only used to perform the task, but can also provide a much more intuitive and easy-to-use environment. In this unit we will examine the common GUIs.

1.4 COMMON GRAPHICAL USER INTERFACES

This section presents a list of terms used commonly with the graphical user interface (GUI). GUIs are systems that allow creation and manipulations of user interfaces employing windows, menus, icons, dialog boxes-mouse and keyboard. Macintosh toolbox, Microsoft Windows and X-Windows are some examples of GUIs.

1) Pointing Devices

Pointing devices allow users to point at different parts of the screen. Pointing devices can be used to invoke a command from a list of commands presented in a menu. They can also be used to manipulate objects on the screen by:

- Selecting objects on the screen
- Moving objects around the screen, or
- Merging several objects into another object.

Since the 1960s, a diverse set of tools have been used as pointing devices including the light pen, joystick, touch sensitive screen and a mouse. The popularity of the mouse is due to the optimal coordination of hand and easier tracking of the cursor on the screen.

2) Pointer

A symbol that appears on the display screen and that you move to select objects and commands. Usually the pointer appears as a small angled arrow.

3) Bit-Mapped Displays

As memory chips get denser and cheaper, bit displays are replacing character-based display screens. Bit-mapped display made up of tiny dots (pixels) are independently addressable and much finer resolution than character displays. Bit-mapped displays have advantages over character displays. One of the major advantages is graphic manipulation capabilities for vector and raster graphics, which presents information in the final form on paper (also called WYSIWYG: What You See Is What You Get).

4) Windows

When a screen is split into several independent regions, each one is called a window. Several applications can display results simultaneously in different windows. *Figure 1* presents a screen with two windows.

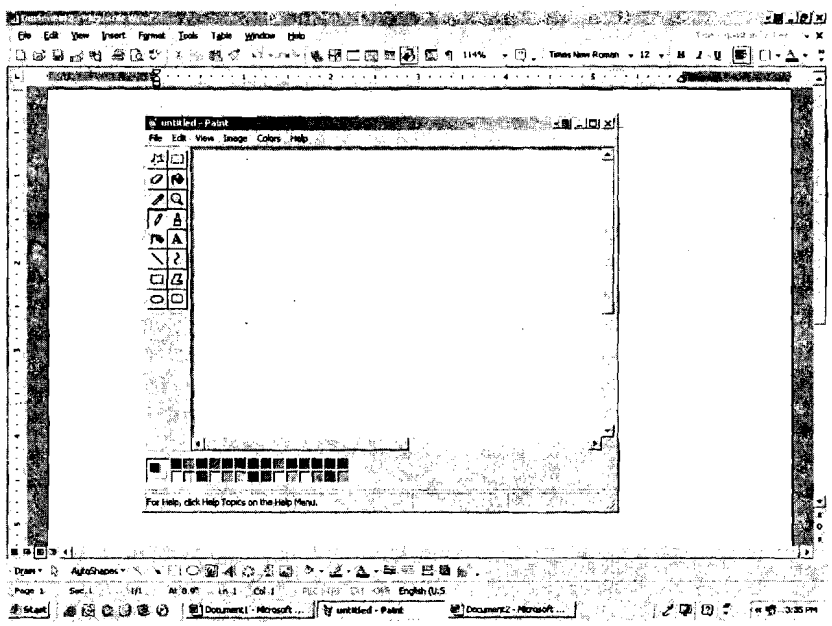


Figure 1: Screen with two windows

The end-user can switch from one application to another or share data between applications. Windowing systems have capabilities to display windows either **tiled or over-lapped,** *Figure 2.* Users can organise the screen by resizing the window or moving related windows closer.

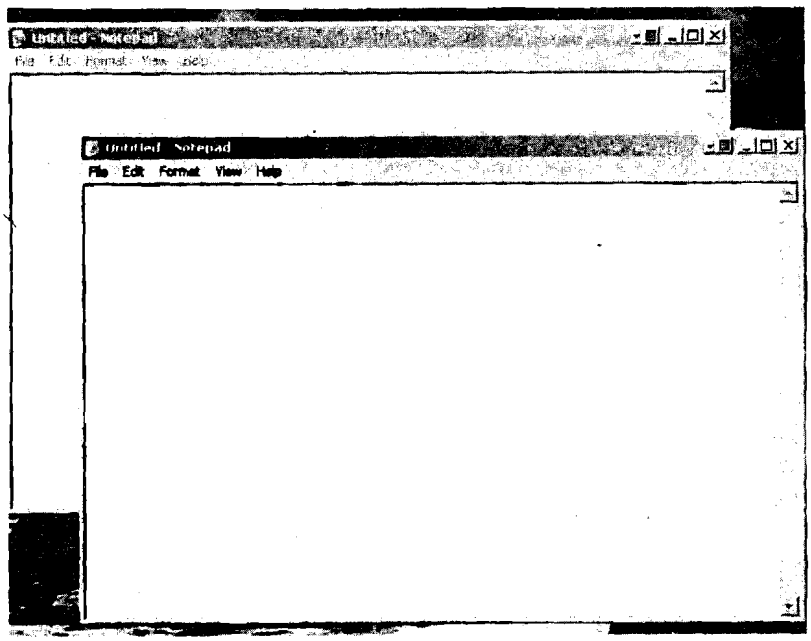


Figure 2: Overlapped Windows

5) Menus

A menu displays a list of commands available within an application (*Figure 3*). From this menu, the end-user can select operations such as File, Edit or Search. Instead of remembering commands at each stage, a menu can be used to provide a list of items. Each menu item can be either a word or an icon representing a command or a function. A menu item can be invoked by moving the cursor on the menu item and selecting the item by clicking the mouse.

Instead of memorising commands to each stage, the user selects a command from a menu bar displaying a list of available commands. For example, *Figure 3* displays the menu bar. This menu bar displays a list of commands available such as File, Edit and the appropriate action is taken.

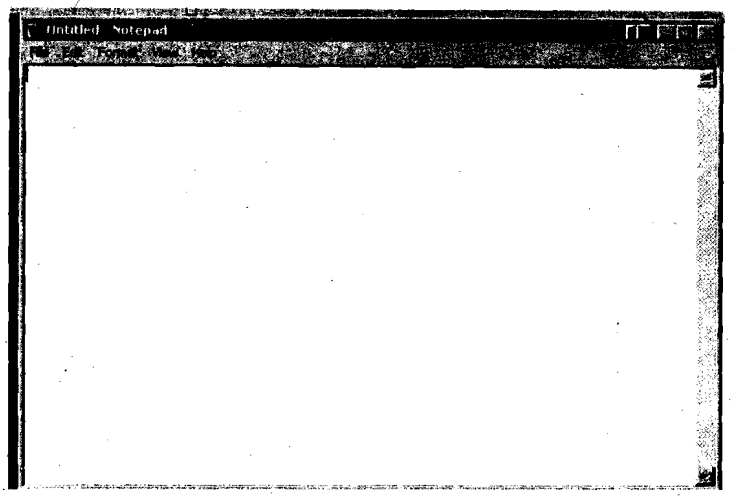


Figure 3: Menu Bar

When a menu item is invoked it could cause other menus, called **pull-down menus**, to appear. **Pull-down menus** (*Figure 4*) are used to present a group of related commands or options for a menu item. *Figure 4* presents the File pull-down menus.

Pull-down and **pop-up** menus display option commands available for each selection. *Figure 4* shows the pull-down menu displayed when the Character menu item is selected. The user can then select from different character styles.

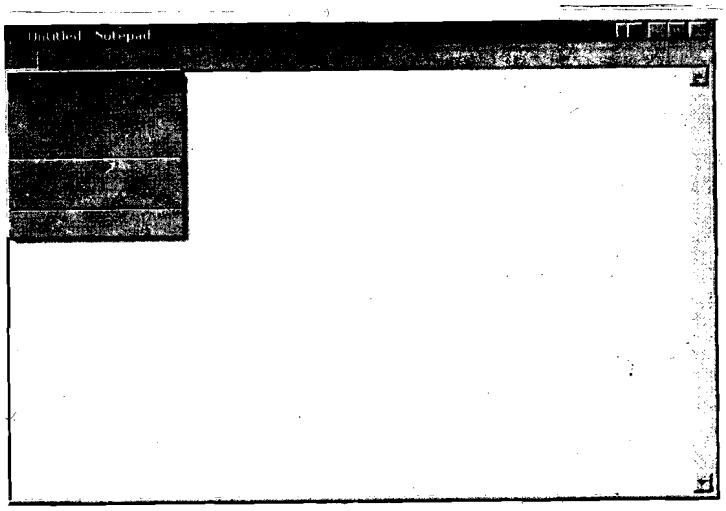


Figure 4: Pull-down Menu

6) Dialog boxes

Dialog boxes (*Figure 5*) allow more complex interaction between the user and the computer. Dialog boxes employ a collection of control objects such as dials, buttons, scroll bars and editable boxes. For example, in *Figure 5*, a dialog box is used to open a file.

In graphical user-interfaces, textual data is not only a form of interaction. Icons represent concepts such as file folders, wastebaskets, and printers. Icons symbolize words and concepts commonly applied in different situations. *Figure 4* shows paint utility with its palette composed of icons. Each one of these icons represents a certain type of painting behaviour. Once the pencil icon is clicked, for example, the cursor can behave as a pencil to draw lines. Applications of icons to the user-interface design are still being explored in new computer systems and software such as the NeXT computer user interface.

Dialog boxes are primarily used to collect information from the user or to present information to the user.

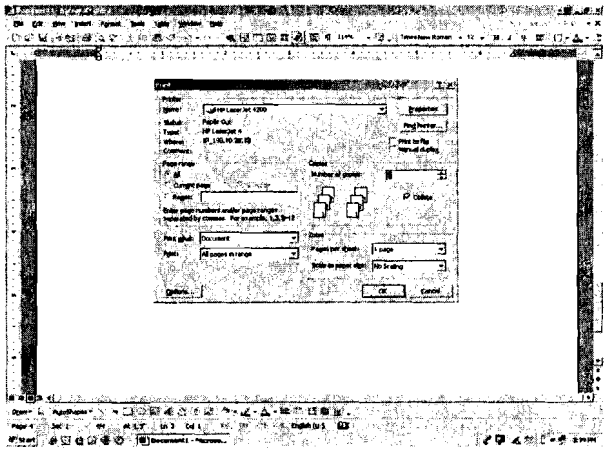


Figure 5: Dialog Box

Among the information obtained are the number of copies and page numbers to be printed. Dialog boxes are also used to indicate error message in the form of alert boxes. Dialog boxes use a wide range of screen control elements to communicate with the user.

7) Icons

Icons are used to provide a symbolic representation of any system/user-defined object such as file, folder, address, book, applications and so on. Different types of objects are represented by a specific type of icon. In some GUIs, documents representing folders are represented by a folder icon (*Figure 6)*. A folder icon contains a group of files or other folder icons. Double clicking on the folder icon causes a window to be opened **displaying a list of icons and folder icons representing the folder's contents.**

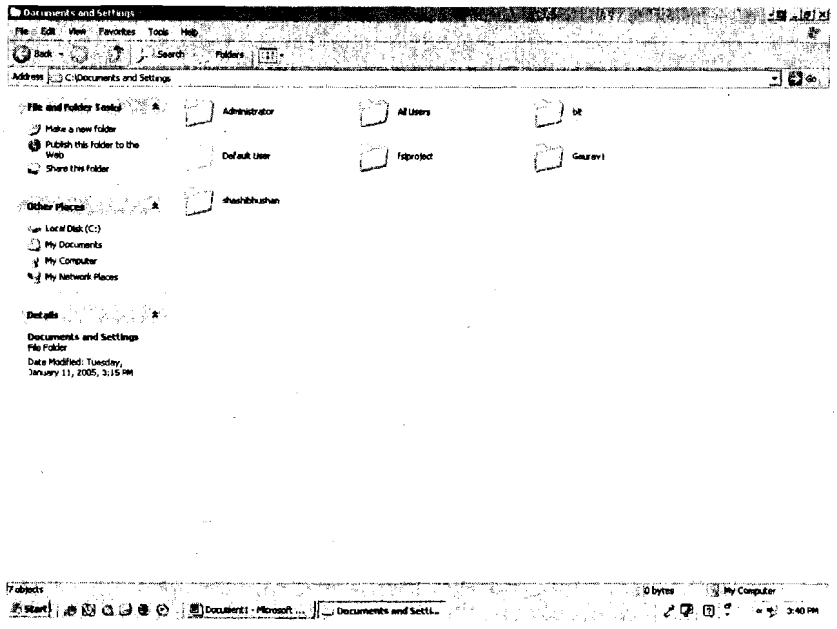


Figure 6: Icons

8) Desktop Metaphor

The idea of metaphors has brought the computer closer to the natural environment of the end-user. The concept of physical metaphor paradigm, developed by Alan Kay, initiated most of the research for graphic user interfaces based on a new programming

approach called object-oriented programming. Discussion of this subject is beyond this unit. The physical metaphor is a way of saying that the visual displays of a computer system should present the images of real physical objects.

For example, the wastepaper basket icon can be used to discard objects from the system by simply dragging the unwanted objects into the dustbin, as in real life. The desktop metaphor probably has been the most famous paradigm. Because of the large set of potential office users, this metaphor can have the most dramatic effect. In this paradigm, the computer presents information and objects as they would appear and behave in an office, using icons for folders, in-baskets, out-baskets and calendars.

In a desktop metaphor, users are not aware of applications. Users deal with files, folders, drawers, a clipboard and an outbox. Instead of starting the word process and loading file, users merely open the report document, which implicitly invokes the word processor. Clicking the mouse on an icon representing the report causes the word processor to get started and to load the report file implicitly. Today, several computing environments provide this capability.

9) The 3D GUI

The desktop metaphor GUI is 2½ D. It is 2D because its visual elements are two-dimensional: they lie in the xy plane, are defined in 2D coordinates, are flat and contain only planar regions (areas). It is 2½ D because where visual elements overlap they obscure each other according to their priority. In a 3D GUI the visual elements are genuinely three-dimensional: they are situated in xyz space, are defined in terms of 3D coordinates, need not be flat and may contain spatial regions (volumes).

The design considerations for a 3D GUI appear more complex than for a 2½ D GUI. To begin with, the issues of metaphor and elements arise afresh. The desktop metaphor with its windows, icons, menus and pointing device elements is firmly established for 2½D GUIs. In contrast no clearly defined metaphor and set of elements for 3D GUIs are manifest — yet. 3D GUIs offer considerably more scope for metaphors than 2½D GUIs; there are many metaphors which could be based on our physical 3D environment, including the obvious extension of the desktop metaphor into a 3D environment, including the obvious extension of the desktop metaphor into a 3D office metaphor. On the other hand, much more abstract metaphors are possible, such as one based "starmaps" where objects are simply placed somewhere in "cyberspace". Likewise the elements of a 3D GUI may resemble, or differ substantially from, the elements of the 2½ D GUI.

The various prototypes have been developed to design the same elements in the 3D GUI as in the 2 $^{1/2}$D desktop GUI: windows, icons, menus, a general space in which to arrange the visual elements, a cursor and an input device to manipulate the cursor.

1.5 FUNCTIONALITY OF GRAPHICAL USER INTERFACES

The development environment for most GUIs consists of four major components:

- A windowing system,
- An imaging model,
- An application program interface (API), and
- A set of tools and frameworks for creating interfaces and developing integrated applications.

Windowing systems allow programs to display multiple applications at the same time. Windowing systems include programming tools for building movable and resizeable windows, menus, dialog boxes and other items on the display. Some GUIs contain proprietary windowing systems, such as Macintosh. Others use common windowing systems such as X-window or simple X.

An **imaging model** defines how fonts and graphics are created on the screen. Imaging models handle, for example, typeface and size in a word processor and lines in a drawing program. This component of the system environment has taken on increasing sophistication as applications incorporate complex curves, color, shading and dimension. Some GUIs support more than one imaging model.

The **API** is a set of programming language functions that allow the programmer to specify how the actual application will control the menus, scroll bars and icons that appear on the screen. Like windowing models, APIs align with particular GUIs.

Finally, **GUI development environments** can include toolkits and frameworks. Most of these toolkits are based on object-oriented approach.

Although the structure of the basic development for most GUIs is similar, there are major differences in how the GUI is integrated with the operating system. Some, like the Macintosh and NeXT GUIs, are closely integrated with the operating system. Others, like X Window or Microsoft's Windows, can be set up as options to be selected by the user when the computer boots up.

Programming of software for GUIs across these components is fairly complex and challenging. Commercial developers who want to support multiple environments find their task further complicated by the absence of standards across heterogeneous computing platforms. The higher-level toolkit component is intended to mitigate much of this difficulty.

Although the graphical user interface has become a standard component of most systems, no standards in windowing systems, imaging models, APIs, or high-level toolkits have emerged. However, three major camps of GUIs dominate. The first camp is IBM's System Application Architecture (SAA), which includes primarily Microsoft's Windows and PM (Presentation Manager). The second camp is UNIX systems, usually built around X Window. The third camp is the Macintosh. In the next section we will describe the object-oriented functionality of representative GUIs in these camps, including Windows, X (upon which most of the UNIX camp is based), NeXT and Macintosh.

☞ Check Your Progress 1

1) What is GUI and what are its features?

.....

.....

.....

2) Define the features of the following:

   a) Window

   b) Pull-down menu

   c) Dialog box

   d) Pointing devices

3) What is the difference between Bitmapped and character based displays?

.....

.....

.....

1.6 GUI DESIGN CONSIDERATION: PSYCHOLOGICAL FACTORS

There are some empirical studies that have identified basic psychological factors that one should consider in the design of a good GUI. In this section we focus our discussion to three primary contributing human factors, which are: the physical limits of visual acuity, the limits of absolute memory, and the Gestalt Principle.

Visual Acuity

Visual acuity is the ability of the eye to resolve detail. This refers to the amount of information one can put on the screen. The retina of the eye can only focus on a very small portion of a computer screen, or anything for that matter, at any one time. This is because, at a distance greater than 2.5 degrees from the point of fixation, visual acuity decreases by half. Therefore, a circle of radius 2.5 degrees around the point of fixation is what the user can see clearly.

At a normal viewing distance of 19 inches, 5 degrees translates into about 1.7 inches. Assuming a standard screen format, 1.7 inches is an area about 14 characters wide and about 7 lines high. This is the amount of information that a user can take in at any one time, and it limits the effective size of icons, menus, dialog boxes, etc. If the user must constantly move his eyes across the screen to clearly focus, the GUI design is causing a lot of unnecessary and tiring eye movement.

Information Limits

Once the user has a desired fixation point, there is a limit to the amount of information that the person can process at one time. A GUI design rule of thumb is that the range of options or choices should never be more than five or six. It has been shown that absolute identification using one-dimensional criteria was about seven items, plus or minus two. Miller introduced the concept of recoding as a method that people use to store information. It has also been pointed out that by expanding the identification criteria from one to more dimensions people could handle more choices and remember more. Later researchers have expanded this work to develop the concept that people chuck information together in order to remember more information. This research has a direct impact on GUI design, especially concerning the number of menu items and icons.

Gestalt Principle

The Gestalt Principle states that people use a top-down approach to organising data. This principle can influence how one should organise graphical information on the screen. The Gestalt school of GUI designers have attempted to identify criteria that cause people to group certain items together in a display. Proper grouping results in a necessary redundancy of selection information that aids the user. For example, if the user knows where one item in a group is on a screen, he will expect other like items to be there also. If one groups the items in line with this expectation, it allows for accurate locating and better transfer of information to the user.

The top-down approach also allows for the development of emergent features. An emergent feature is a global property of a set that is not evident when one views each

item locally. Since global processing tends to be automatic, one can argue that an emerged feature reduces the attention demand as a user operates a multi-element display. For this performance enhancement, one must use the Gestalt Principle in the initial placement, and the resulting organisation must be compatible with the user's cognitive view of the task.

1.7 GUI DESIGN CONSIDERATION: STANDARDS

Considering the above psychological factors, one could come to the conclusion that one could easily extrapolate these factors to the design of a good GUI. Empirical studies of GUI show that this intuition is not always the case. It directly leads to the conclusion that a good GUI would use a lot of icons. Unfortunately, too many randomly placed icons violate the limits of absolute memory. Using the Gestalt Principle, one can group like items together using factors like color to add more informational dimensions. Too many colors, however, destroy the global visual grouping of the items. The user then begins to concentrate on the GUI. Any primary cognitive task attention devoted to the interface may interfere with the primary task. One can derive basic GUI standards from basic human factors, however. The standards are the presentation of information, the grouping of information, and information sequencing.

Amount of Information Presented

The amount of information to present is the most basic of GUI design considerations. H.E. Dunsmore showed that making screens less crowded improves screen clarity and readability. As such, GUI designers usually follow the guidance that the interface should display only what the user needs to perform the current operation. Empirical researchers show that limiting the information to that necessary for the user reduces errors and time to perform tasks. Errors and performance time increase as the GUI presents more information. Of course, it requires a thorough analysis of the tasks that the user must perform in order to display only the necessary amount of information.

Compared to a randomly placed screen, a well-designed screen can reduce the time needed to perform a task by as much as 40%. Ways to conserve screen space are:

1) Appropriate use of abbreviations: Many design documents recommend using complete words whenever possible. Due to screen sizing constraints, it is not always possible to use complete words. When complete words are not possible, abbreviations should be contextual and consistent. A good contextual example is "h," which is usually a good abbreviation to use for help. The number of abbreviations should not only be contextual but also be kept to the minimum. As a poor example, in the UNIX system, the "Is" command lists files in a directory. The "Is" command has 17 different one-letter abbreviations that change the output options of the "Is" command. The one-letter abbreviations have little contextual link to the options they represent. In fact, the UNIX system is a good example of what not to do.

2) Avoid unnecessary detail: For example, use whole numbers if one does not need decimals. Keep the window and icon designs clear and simple. Even when users prefer more complex icons, elaborate icons add nothing to performance. Studies show that when icon designs are too complex, time to complete a task actually increases. In studies with 3-D and 2-D graphical displays, users preferred the 3-D displays. There were no differences in performance between the two graphical displays, however.

3) Use concise wording: Screens have limited space. Screen designers should avoid the tendency to place additional data on the screen just because the data is available. More objective limits of screen density vary from the thresholds of

25% to 80%. There is no empirical research that substantiates any performance enhancement with any specific threshold.

4) Use familiar data formats: With more familiar formats, the user will need less information to complete the task. An example for data entry is the standard USA address format of street, city, state, and zip code. In addition to requiring less instruction, the user will perform the operation faster than if the format is unfamiliar.

5) Use tabular formats with column headings: Tabular formats allow for efficient labeling of related data. It is especially preferable for data location tasks. Simply splitting items on one long line into two lines result in productivity improvements of 20%.

Grouping of Information

Given a set of information to display, there are many ways one can display the information. Proper grouping improves the information's readability and can highlight relationships between the information.

There are several techniques to aid in the grouping of information, which include:

1) Color: Presenting different groups in different colors clearly creates some degree of grouping among the elements of the same color. GUIs that utilise color well increase productivity. If like color items are in close proximity, the visual association is stronger than if the like color items are further apart. In addition to changing the item's colors, one can use different colors for the background and foreground. The effectiveness of this technique decreases as the number of screen colors increases. Overuse of color degrades performance, however.

2) Graphical Boundaries: Drawing boundaries around elements is the most common method of grouping elements in GUI. Although there is no empirical evidence to show that these groupings improve performance, users prefer this type of groupings compared to other methods. Another method of grouping is to group tasks within icons. Icon grouping is easy because many icons can have common attributes. Icons are also small and therefore use less space. Another advantage of icons is that recognition is faster for pictures than for text. This makes it easier for the novice to learn a system. Studies also show that icons have smaller error rates than textual interfaces and the same as for menu interfaces. Conversely though, empirical studies have shown that, counter intuitively, icons do not lead to greater increases in performance.

3) Highlighting: Besides color, there are several other methods of highlighting including reverse video, brightness, underlining, and flashing. The most common use of highlighting is reverse video to indicate an item that is currently selected. GUIs usually use brightness to show which items are not active at a given time. Underlining is effective if it does not interfere with the legibility of characters. Flashing will both get attention and annoy if the user cannot turn off the flashing. Therefore, one should use flashing only to convey an urgent need. Apple Macintosh uses flashing to signal only program or data destruction. Regardless of which type of highlighting, one needs to apply it conservatively. Overuse of highlighting causes confusion among users and defeats its purpose. Additionally, if one highlights the wrong information, the user has more difficulty detecting the important information.

Information Sequencing

One needs to lay out a screen in a manner that allows the user to easily find any information on it. Most designers advocate the use of the de facto GUI screen

standards. This is because many users now expect certain modes of operation in all GUIs. For example, most users expect the top of screen to contain the headings for the pull-down menus. The top right is the default location for icons representing the disk availability. In the Macintosh GUI, the bottom right contains the trash icons used for deleting files.

Within a window, there are also many standard modes. A window title is usually at the top. Scroll bars are on the right and bottom for vertical and horizontal window movement. A box for closing the window is at the top left. Icons for resizing the window are at the four corners.

Studies show that most users initially scan the screen starting at the upper-left corner. This corner should be the obvious starting point for applications invoked from within the window. This permits a left-to-right and top-to-bottom reading, which is standard for Western cultures.

The optimum sequence for screen presentations is a collection of various factors, including:

1) Sequence of use: One needs to present the user the information in the order that the user will probably utilise it.

2) Conventional Usage: If a common convention is in general usage, the GUI design should continue using it. For example, in the standard window layout, the file option is usually to the far left of the menu bar.

3) Importance: The designer needs to place the more important information at a prominent location. For example, if several entries are possible, the GUI should lead off with the required ones and end with the optional ones.

4) Frequency of use: One should pace the most frequently utilised commands at the beginning. For example, in a menu list, the most frequency utilised commands should be at the top of the list.

5) Generality versus Specificity: The more general items should precede the more specific items, especially when there is a hierarchical relationship among the data.

6) Alphabetical or Chronological: If there is no other rules for ordering data element, then one should adopt some other technique such as an alphabetical or a temporal listing. Card [11] showed that selection time was faster for alphabetical than for any other functional grouping.

1.8 GUI EXAMPLES

The goal of any GUI is to allow the user to work through the computer and application to concentrate on the primary cognitive task. The user should not be concerned with the user interface. Any attention devoted to the interface interferes with the main task.

This section presents Graphical User Interfaces. Some popular GUIs will be covered to provide a broad picture of the subject. There are a great many popular GUIs around including X Windows, Microsoft Windows, NeXT's NeXTStep and others.

1.8.1 Microsoft Windows (MS-WINDOWS)

MS-Windows is the most popular GUI for IBM personal computers. IBM and Microsoft announced OS/2 as a new operating system for 80286 and 80386 personal computers. The OS/2 Standard Edition 1.1 adds Presentation Manager (PM) for its graphical user interface. The user interfaces of Windows and PM are very similar but their APIs are different. Microsoft's strategy is to use Windows as a springboard for PM.

Windows provides an environment that enhances DOS in many ways. The major benefits of Windows are:

1) **Common Look and Feel:** All windows applications have the same basic look and feel. Once you know one or two Windows applications, it is easy to learn another one.

2) **Device Independence:** Windows presents a device-independent interface to applications. Unlike most of today's DOS applications, a Windows application is not bound to the underlying hardware such as mouse, keyboard or display. Windows shields the application from this responsibility. The application deals with the Windows API to manipulate any underlying devices.

3) **Multitasking:** Windows provides non-preemptive multitasking support. Users can have several applications in progress at the same time. Each application can be active in a separate window.

4) **Memory Management:** Windows also provides memory management to break the 640K limitation of MS-DOS. An application has the ability to use the extended memory, share data segments with other applications and unwanted segments to disk.

5) **Support for existing DOS applications:** Windows allows most standard DOS applications to run under it directly. Any application that does not control the PC's hardware, use the PC BIOS or MS-DOS software interrupts, can run in its own window.

6) **Data Sharing:** Windows allows a data transfer between application Clipboard. Any type of data can be transferred from one window with the clipboard. The Dynamic Data Exchange (DDE) protocol defines how two applications can share information. Information such as bitmap, metafile, character strings and other data formats can be shared.

Support for Object Orientation

In order to create screen objects such as windows, the application developer defines a class (similar to record) specifying the necessary properties. Instances of class can then be created. Several applications can share the same windows simultaneously. To communicate with Instances of a window class, messages are sent and received by a special function called the window function. The windows handle all messages such as re-drawing the screen, displaying icon or pop-up menus and changing the contents of the client area.

Creation and Manipulation of a Window

MS Windows presents a pre-defined style for user-defined windows; it presents the structure of a window, followed by a discussion of windows manipulation.

1) **Structure of a window:** *Figure 7* displays possible elements for a window. The caption bar (or title bar) displays the name of application within the window. The system menu box contains names of commands available in all applications, such as minimise, maximize, resize and close. The minimize box clicked once reduces the window to an icon. The maximize box enlarges the window to the full screen. The menu bar contains a list of commands In the application. The client area is the area inside the window, which is under the application control.

2) **Creating windows:** Each window created on the screen is a member of some user defined window class. Window classes are created by the application program. Several window classes can be active simultaneously. Each window class in turn can have several instances active at the same time. There are no predefined generic window classes that come with MS-Windows.

To create a window the following steps must be taken:

a) **Set up a window class structure** which defines the attributes of the window class. Attributes that can be defined include:

i) the window function, which handles all messages for this class

ii) the icon and the cursor used for this class of windows

iii) the background color of the client area

iv) the window class menu

v) the re-drawing function used when resizing horizontally or vertically.

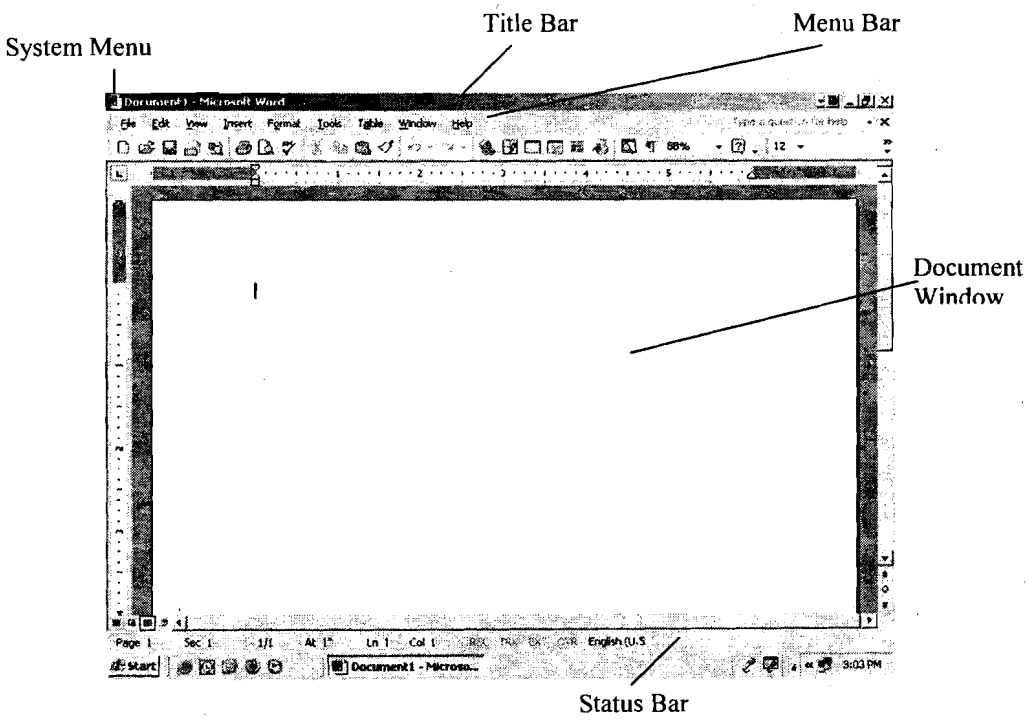


Figure 7: MS-WINDOWS screen element

3) Manipulating windows: An application can choose to display the Windows, resize the window, display additional information in the client area, and so on.

Pop-up and Child Windows

Pop-Up and child windows are special type of windows, and are used to communicate information between the application and the end-user. They remain on the screen for a short period of time. In this example, the pop-up Display information about a given file such as date and time of creation and the size. Dialog boxes, as discussed earlier, are a more sophisticated form of pop-up windows.

MS-Windows provides a collection of predefined child windows. These are the most common usage of child windows. These predefined classes are button, scroll bars, listbox, edit and static class. A developer can also define child windows that can be controlled by any user-defined operation.

Resources

Resources are used to manage windows and user-defined object. **MS-WINDOWS** provides nine kinds of resources to application developers. These resources are : icons, cursors, menus, dialog boxes, fonts, bitmaps, char strings, user-defined resources, and keyboard accelerators.

1) **Icons and cursors:** Windows defines a few types of icons and cursors. An icon or a cursor is essentially a bit-mapped region that is used to represent and symbolise a window or cursor. A developer can also define an original icon or cursor using the ICONEDIT utility.

2) **Menus:** Each window can have its own menu bar. A menu item can be a character string or a bitmap. Each item of a menu bar in turn can have a pop-up menu presenting a list of options. Currently, Windows does not support nesting of pop-up menus within other pop-up menus. (Windows 3.0 provides this functionality). But a pop-up menu can invoke a dialog box. When a menu is selected, Windows sends one or more messages to the Window function of the Window containing the menu bar. These messages can be interpreted to perform the function corresponding to the menu item.

3) **Dialog boxes:** These provide another mechanism besides pop-up menu and menu bar to obtain information from the end-user. Dialog boxes are much more flexible than menu bars or pop-up menus. Dialog boxes usually contain a group of child windows such as buttons, scroll bars, and editable fields. Just like windows, dialog boxes have a function that is used to process messages received from the user upon selection of options. Generally, dialog boxes appear as pop-up windows. The user selects the option needed from the dialog box and the dialog box disappears. *Figure 8* depicts an example of a dialog contains an edit box, a list box, and open and cancel buttons. The end-user can specify the name of a file either by selecting from the list box or by typing the name of the file in the edit box. By clicking on the open button, the application will open the selected file.

4) **Fonts:** Windows provides a few families of fonts with different sizes and shapes: Modern, Roman, Swiss, Helvetica, and Script. Application processors and desktop publishing can define additional fonts as needed.

5) **Bitmaps:** They are used to represent icons, cursors, or draw picture on the screen. Both mono and color bitmaps can be defined.

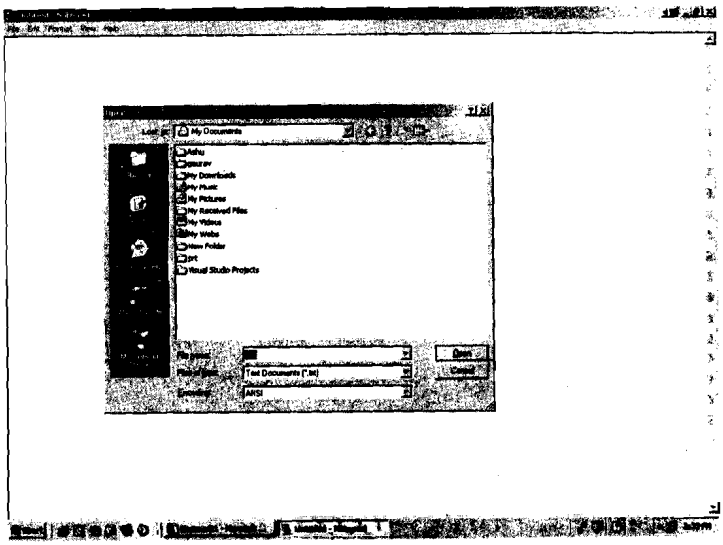


Figure 8: Dialog Box (MS-Windows) with an edit box, a list box, and push buttons

6) **Character strings:** Character strings are handled as resources mainly to provide a manageable solution to internationalisation of a window application. Instead of including a character string as part of the code, it can be placed in a resource file handled as a resource. Once in a resource file, different versions of

the same character string resource file can exist for different languages. This separation of the character strings from the code makes internationalising the application much easier, and also makes maintaining the application much simpler.

7) **User-Defined Resources:** These can be used for many purposes and support any user-defined data type. Any arbitrary data can be managed as a user-defined resource.

Resources are defined in a text file called a resource script. They are compiled through a utility called the Resource Compiler and linked with the windows application. Resources are read only data. Once a resource script is compiled, it can be shared with other window applications.

Graphics Device Interface

Graphics in Windows are handled by the Graphics Device Interface (GDI). Both vector and raster color graphics are supported by GDI. GDI supports only two-dimensional graphics. Vector graphics are supported by a set of vector drawing functions such as drawing a line, point, and polygon etc. pie chart.

Raster graphics are supported by pixel manipulation. Raster graphics can be stored or manipulated either as a bitmap or as a metafile. A bit-mapped representation of the graph can be manipulated using utilities such as BitBlt, PatBlt, and StretchBlt. A metafile provides binary encoding of GDI functions such as to draw vectors and to fill a region with a bitmap. Metafiles take up less disk space than a bit-mapped representation since they do not represent each pixel directly on disk. When metafiles are played, they execute the function encoding and perform the necessary graphics operations to display the graphics output.

☞ **Check Your Progress 2**

1) What are the four major components of GUI? Explain the functioning of any one component.

.....

.....

.....

2) How does MS-Windows enhance DOS environment?

.....

.....

.....

3) How are graphics supported in MS-Windows?

.....

.....

.....

1.8.2 Macintosh Toolbox

The tremendous success of the Macintosh computer popularized the window-style menu-driven user interface. Macintosh got its early start from Apple's Lisa.

The Macintosh GUI is called the Toolbox. The Toolbox consists of a collection of utilities to manipulate Macintosh's resources. In this section we present an overview of Toolbox functionally, examine the object-oriented features of the Toolbox and discuss major components of the toolbox's user interface features.

Functional Overview

The Toolbox provides a collection of utilities to access and manipulate Macintosh hardware and software resources. It provides a set of utilities to manipulate interface components such as windows, menu bar, and dialog boxes. These are discussed in the following sections. Some of the other utilities provided are:

1) **Fonts Manager:** allows manipulation of system and user-defined fonts.

2) **Event Manager:** provides monitoring of events generated by keyboard and keypad.

3) **Desk Manager:** provides access to desk utilities such as the calculation.

4) **Text Edit:** provides simple text editing capabilities such as cut and paste.

5) **Memory Manager:** provides a set of routines to manipulate dynamic memory.

6) **File Manager:** provides a set of routines to manipulate files and transfer data between files and applications.

7) **Silver Driver:** provides access to sound and music capabilities.

8) **Toolbox Utilities:** provides a set of general routines to manipulate icons, patterns, strings, fixed-point arithmetic, and so forth.

The Toolbox can be manipulated either from the assembly language or Macintosh Programmer's Workshop (MPW) language (Pascal, C, C++). Toolbox can be accessed from non-MPW languages including Think C. One of the major features of the Toolbox is that most of the Toolbox utilities reside in ROM. This provides a very responsive user interface.

From this point on, we will concentrate on the user components of the Macintosh Toolbox such as the Window Manager, the Manager, the Menu Manager, the Dialog Manager, the Scrap Manager, the Draw, and the Resource Manager.

Object-Oriented Features of Toolbox

Just like Microsoft Windows, the Macintosh Toolbox design is object-oriented. For example, the application developer defines a new type of window by the template for a new class of windows. Messages are sent between Toolbox application to change the behaviour of screen objects such as windows and cursors. When a window needs to be drawn or resized, a message is sent to the **window definition function** to perform the appropriate action.

The Window Manager

The Window Manager allows the application developer to manipulate windows on the screen. It deals with issues such as overlapping windows, resizing windows, moving windows around the screen, or changing a window from a foreground to a background window. Developers can create and manipulate pre-defined Toolbox windows with any desired shape and form. For example, the user can create a circular or a hexagonal window.

Toolbox comes with a series of pre-defined windows. The most popular of these windows is the document window. The document window, depicted in *Figure 9,* has

the following regions: Title bar, Close Box, Scroll bar(s), Size Box and content region. The Title bar contains the title of the window. By clicking and holding the mouse within the region, the window can be dragged to a different screen. The Close Box is used to shut the window. Clicking inside this box will cause the window to disappear. The Size Box is used to resize the window. The horizontal or vertical Scroll bars can be used to scroll the contents of the window. The content region is the area that is controlled by the application.

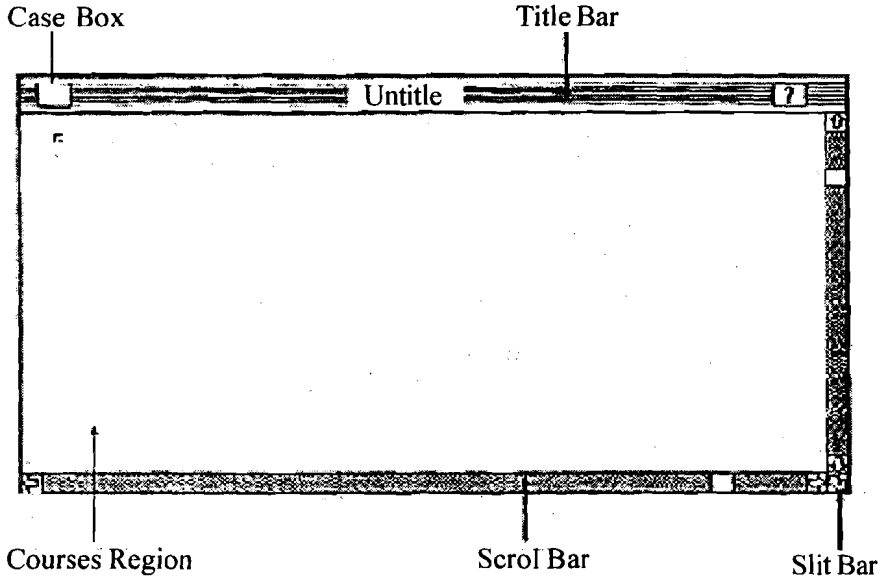


Figure 9: Macintosh Toolbox Document Window

When creating a document window anyone of the components described above can be omitted. For example, anyone of the Scroll bars can be absent from the document window.

New types of windows can be created through either the Window Manager or the Resource Manager. The Resource Editor can be used to create a window template. A window template essentially defines a class of windows that can be instantiated by an application. For each class of window, defined by the system or there exists a function called the **window definition function**. This function is used to handle the behaviour and appearance of the window. When a window needs to be drawn or resized, a message is sent to the **window definition function** to perform the action for that type of window.

The Window Manager provides a large collection of routines to create, display, hide and activate/deactivate windows on the screen.

The Resource Manager

The Resource Manager provides access to various resources such as the windows, fonts, icons and menus. A user-defined resource such as a window is defined as Resource Editor. The template of the window and the window definition function is stored in a file called the resource file. A unique resource identifier is used to access a pre-defined resource. Resource identifier can be used to recognise the type of resource and the actual resource file to be read.

The Menu Manager

The Menu Manager provides a set of routines to create and manipulate menus. A menu bar, depic ted in *Figure 10,* can contain a list of one or more menu items. Each menu item highlights itself when clicked with the mouse; this item is selected. The developer can define a menu using the standard bar. Menus can be stored as a resource and managed by the Resource Manager. Once given to Manager, a unique identifier is used to reference the menu.

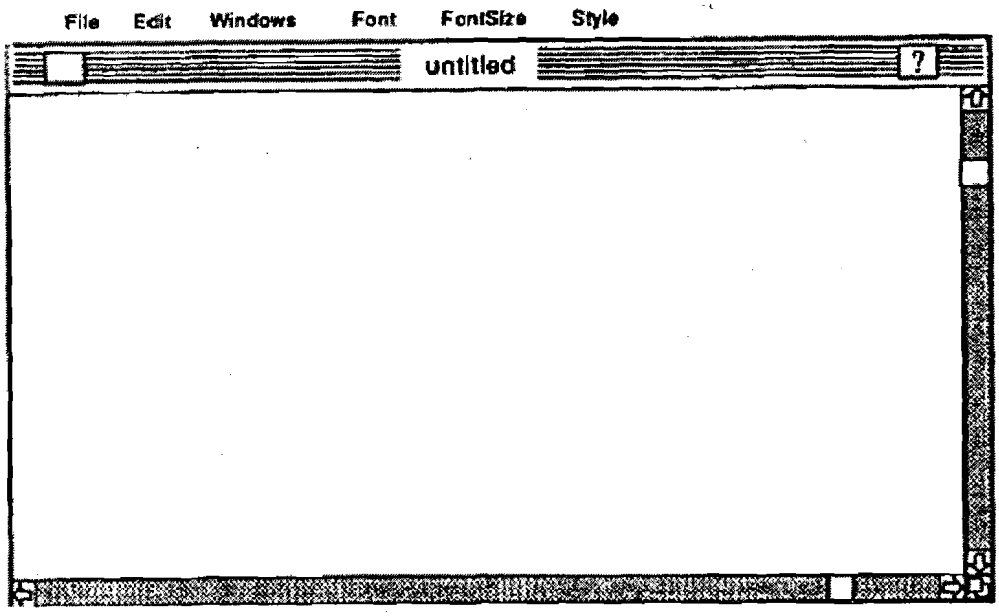


Figure 10: Macintosh Menu Bar

A menu item can even invoke a pull-down menu, which can be used to or choose additional selections, see *Figure 11*. A menu item in a pull-down menu can then invoke a dialog or an alert box. A menu item can be a textual icon. Unlike MS Windows, menus in Toolbox cannot be a part of a window Menus and only used to define the menu bar, which is associated with one Macintosh screen.

The Control Manager

The Control Manager provides a set of routines to define objects like buttons, check boxes and scroll bars. Controls are usually a window. For example, the scroll bars can be defined as part of a document. Most often controls are defined as part of a dialog or an alert box.

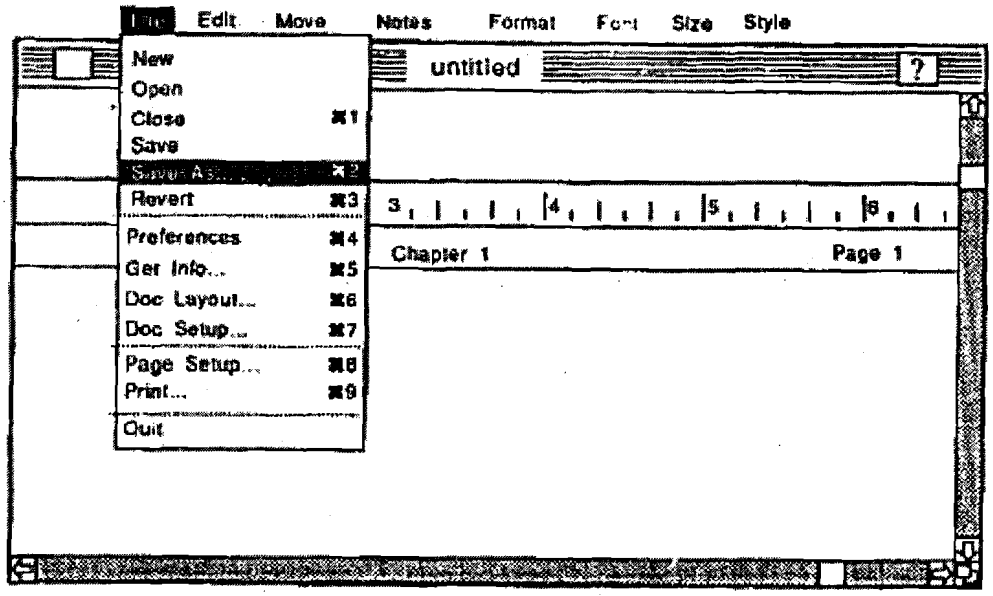


Figure 11: Macintosh Pull-down Menu

The Dialog Manager

The Dialog Manager provides a set of routines and data structures to manipulate dialogs and alert boxes. Dialog boxes are used to get additional information from the end-user of an application. An alert box is used to provide information such as a warning or a cautionary message.

The Scrap Manager

The Scrap Manager allows the user to move data from one application to another. A portion of one window can be cut and placed in the scrap manager and then pasted into another window. Once a portion of a window is cut, it is placed in the clipboard. The user can paste the contents of the clipboard into another window.

1.8.3 X-Windows

The name X, as well as part of the initial design, was derived from an earlier window system called W developed at Standford University. Currently, the X Window system is supported by an X Consortium of primarily UNIX-based hardware and software vendors as a standard base for user interfaces across most UNIX product lines. *Figure 12* illustrates the components of UNIX GUIs based on X Windows.

The X Window system does not define any particular style of interface but rather provides a mechanism for supporting many styles. X is also network-based. In contrast with PM, the X architecture is based on the premise that an application can run one computer, while the graphical presentation of the application's output and responses from the user can occur on another computer. As such, X provides a flexible set of primitive window operations but carefully avoids dictating the look and feel of any particular application. X's device-independent functionally allows it to serve as a base for a variety of interface styles. X does not provide user interface components such as buttons, menus, or dialog boxes.

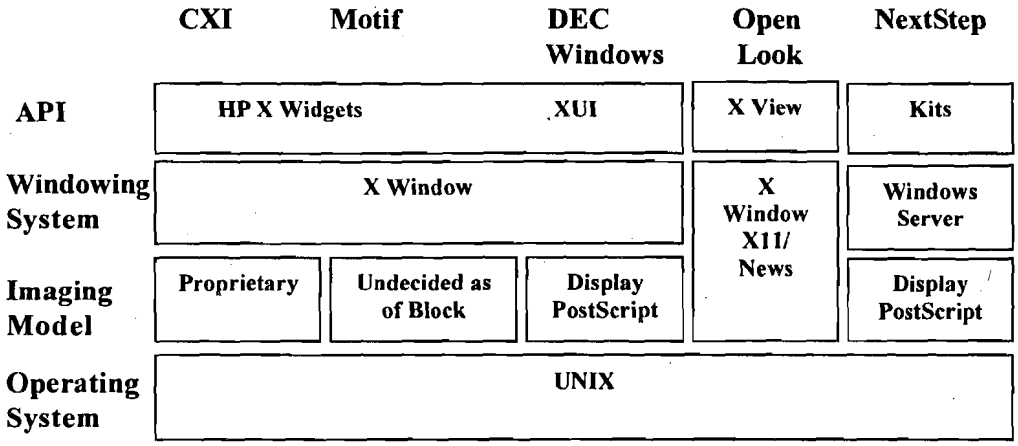


Figure 12: Components of major UNIX-based GUIs

An interface supporting X can theoretically use any X-Window display. The application program sends calls to the X-Window Library, which packages the requests as X packets and sends calls to the X-Window server. The server decodes the X packets and displays them on the screen. Exactly how the packets will be displayed (i.e. will look) on a workstation depends on the set of pre-designed window elements called widgets that are unique to the particular workstation in the system environment. The ultimate look on one workstation may differ substantially from that on another, but the response to a user's action on these different-looking interfaces will be the same. Several hybrids of X exist.

The application-programming model for X, like PM and Windows, is event-driven, and the categories of events are similar. Like PM, X programs rest in wait loops until the underlying window-management system generates an event. Window hierarchy management is also common to the architecture of both systems. Like PM's window, windows in X are organized as parents and children, so attributes can be inherited and effects of the events applied to related members.

Since X does not support any particular interface style, most programmes build applications with libraries that provide an interface to the base window system. One standard programming interface to X is the C language library known as X Library or Xlib. This library defines functions for access and control over the display, windows and input devices. Although commonly used, the library can be difficult and tedious to work with and often results in hundreds of lines of code for a GUI. A better solution is a higher-level toolkit, such as the X Toolkit (Xt), upon which many other X Window toolkits are based.

X Toolkit is the foundation upon which most of the commercial toolkits are based and is a good example for describing object-oriented functionally. The X Toolkit consists of two parts: a layer known as Xt Intrinsics and a set of widgets. Xt Intrinsics supports many of the available widget sets. The widget set implements user interface components including scroll bars, menus and buttons. Xt intrinsics provides a framework that allows the programmer to combine the components. Both parts are integrated with Xlib, allowing programmers access to higher-level libraries. A typical X programming environment is illustrated in *Figure 13.*

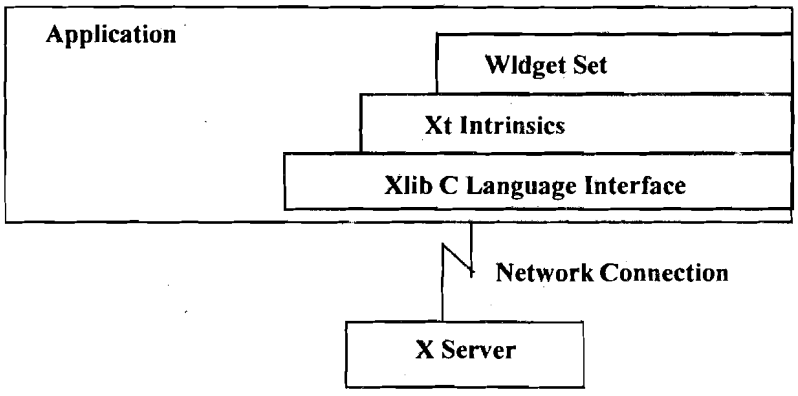


Figure 13: Typical X Window Development Environment

The common distinction in object-oriented programming between an application programmer (consumer) and a class library programmer (producer) fits well into the X environment. The X application programmer is likely to use a combination of a higher-level toolkit (e.g., the Toolkit's Xt Intrinsic), Xlib, and a widget set. The class library programmer, whose job is to create new widgets, is likely to use the capabilities from within one of the higher-level toolkits.

Depending on the toolkit, the widget programmer may be able to take advantage of object-oriented capabilities. Xt Intrinsic, for example, uses an object-oriented approach and organises widgets into classes. Xt Intrinsic defines the basic architecture of a widget. This allows widgets to work together smoothly when built by different programmers or potentially when selected from different widget sets.

A typical widget consists of two basic parts: a class record and an instance record. Each of these components is implemented as a C structure containing data and pointers to methods. Intrinsics defines the organisation of each structure. All widgets belonging to a class share a copy of common data methods for that class. Each individual widget has its own copy of instance-specific data. A widget's class record is usually allocated and initialised statically at compile time; a unique copy of the instance record is created at run time for each individual widget.

Since all widgets belonging to the same class share the same class record, the class record must contain only static data that do not relate directly to the state of an individual widget. For example, every widget's class record includes a field containing the widget's class name. The class record also contains methods that define the

appearance and behaviour of all widgets in the class. Although most of these methods operate on the data in the widget's instance records, the methods themselves are shared by all widgets in a class.

Many object-oriented languages provide inheritance as a language construct. The Xt Intrinsics is written in the C language, which does not directly support object-oriented programming. Xt itself supports inheritance with a subclassing mechanism. To use Xt's subclassing capabilities, the programmer first finds an existing widget with similar functions (a common step in object-oriented design), writes a subclass that can inherit existing data and methods, and then adds new data and methods as needed.

If a similar widget cannot be found, then a foundation class called Core is subclassed. All widget classes are subclasses of the core widget class. Like the class Window in PM, Core contains the basic methods for initializing, displaying, and destroying widgets, and reacting the external resizing. Core also stores basic properties (e.g., the geometry) of widgets and the data for handling events.

New widget classes inherit the methods (called resources) defined by their superclass by specifically including the definition of the superclass structure in the definition of the new class. Xt Intrinsics provides two mechanisms for inheriting the methods defined by a superclass. The first mechanism is referred to as chaining. When a method is chained. Xt Intrinsics invokes the method defined by a widget's superclass first and then invokes the widget's method. This allows a widget to inherit part of a method from its superclass.

Xt Intrinsics also provides a mechanism for inheriting methods that are not chained. This is done by using special symbols to specify methods in the widget's class records. Each symbol is defined by the superclass that added the method to the widget's class record. These symbols can be used by a subclass of the widget class Core and do not have to be redefined by each widget class. Only classes that contribute new methods to the class record need to define new symbols. When a widget class specifies one of these symbols in the class record, Intrinsics copies the corresponding method used by the widget's superclass into the widget's class structure at class initialisation time. *Figure 14* illustrates this architecture and shows the relationship between the class record and instance records of several widgets belonging to widget class core.

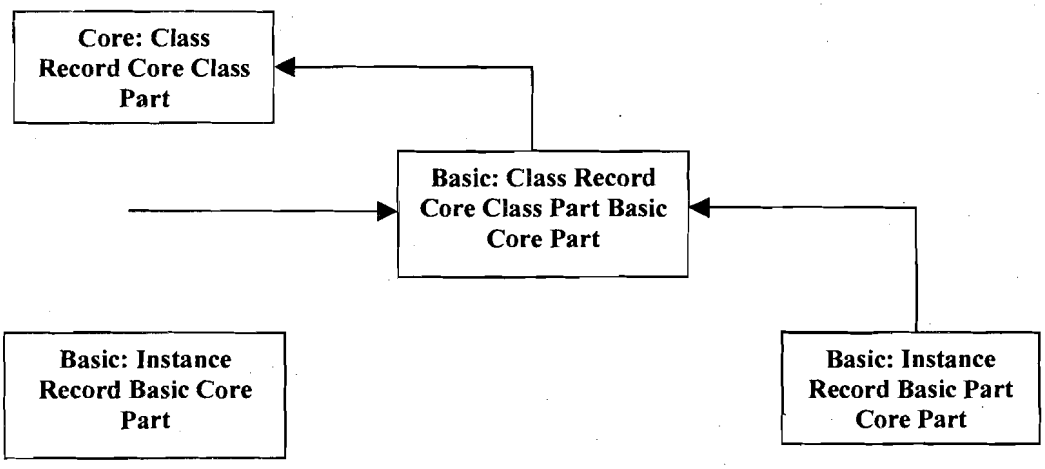


Figure 14: Inheriting from Widget Class Core

The Xt Intrinsics uses a data-abstraction technique to hide the implementation of a widget from applications that use the widget. Applications that use a widget see only the incomplete definition of the widget and therefore cannot directly access fields in the widget structure.

Applications declare all widgets as type Widget, which is known as opaque type. This means the application has a pointer to the widget structure but does not have access to the real definition of the data that it represents. The application cannot access the contents of the data structure.

Another object-oriented technique promoted in X Windows is to extract some general functionality from two or more new widget classes and create a metaclass. A metaclass is an abstract class that is not intended to be instantiated directly but serves as a superclass for other similar widgets. When creating a complete new widget set, it is frequently useful to create a class of hierarchy or metaclasses. With this approach, the top metaclass in the hierarchy defines elements that all widget classes in the set have in common, and each subclass becomes more and more specialised. Such an organisation allows the widget programmer to create new widget classes with the least amount of effort, although there is some extra initial effort required to design and create metaclasses. For example, this approach is used by the X Widget set from HP, where most basic widgets inherit from the primitive metaclass and most composite widgets inherit from the Manager widget class.

☞ Check Your Progress 3

1) What types of utilities are provided in Toolbox? Explain their features.

.....

.....

.....

.....

2) Explain the functioning of Resources Manager and Menu Manager of Toolbox.

.....

.....

.....

.....

3) What is the basic philosophy of X-windows? How is it different from the rest of GUIs?

.....

.....

.....

.....

1.8.4 NeXT

The NeXT computer, with its three-dimensional user interface, was introduced in 1988. It has grabbed the attention of the computer industry. The machine has been hailed as the most innovative computer invented in recent times. The computer was initially intended for the educational market. But the NeXT Corporation decided to widen the market for its machine to the commercial arena.

We provide a brief overview of the NeXT software tools and capabilities, followed by a discussion of the capabilities of the user-interface design.

Overview of NeXT Software

The NeXT computer is designed to reach a wide range of users from non-technical to power users. The non-technical users can deal with the graphic user interface to perform

tasks by manipulating menus, icons, and dialog boxes. The power user can directly interact with its Mach Operating system.

The NeXT system software comprises three major pieces: the Mach operating system, applications and the NeXT user interface. The Mach operating system, developed at Carnegie Mellon University, is a re-designed UNIX. Mach re-designs the UNIX kernel to reduce the size.

NeXT also comes with a set of bundled applications. Currently, there are new applications being developed that take advantage of NeXT hardware and software capabilities. Some of the applications supported on NeXT are NeXT SQL Database Server, Mathematica (symbolic mathematics package, WYSIWYG editors and word processors, and more.

NeXT User Interface

The third and last component, the NeXT user interface, is the most impressive piece of NeXT technology. The NeXT user interface draws heavily on direct manipulation and modern graphic user interfaces. The NeXT user interface is composed of four components. Workspace Manager, Interface Builder, Application Kit, and NeXT Window Server.

The Workspace Manager allows the user to manage files and directories and to execute programs. When a user logs into a NeXT machine, the Workspace Manager is started, *Figure 15.* The Directory Browser window is used to navigate through the files on disk.

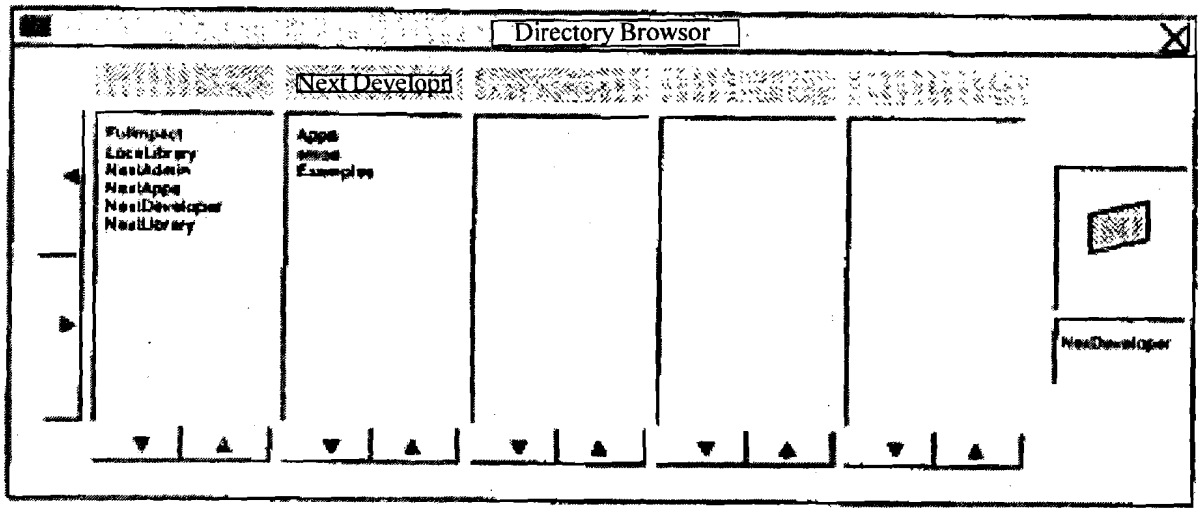


Figure 15: NeXT Workspace Manager Directory Browser

The Interface Builder lets the user create interfaces on the screen without writing a single line of code. Users simply select the menu, control and screen objects from a palette, and then move the controls to the desired location.

The Application Kit is a library of user-interface objects. It is used in conjunction with the Interface Builder to design user interfaces. The Application Kit is discussed in the next section.

The Window Server handles all screen activities, such as drawing windows and handling events such as mouse clicks. The window Server itself does not perform the drawing and screen I/O commands. Display PostScript, designed by Adobe and NeXT, handles all such activities. Up to now, PostScript has been used only as a language for printer engines. With the advent of the PostScript, both screen and printer share the same protocol. Therefore one drawing method is used to display objects on the screen and the printer.

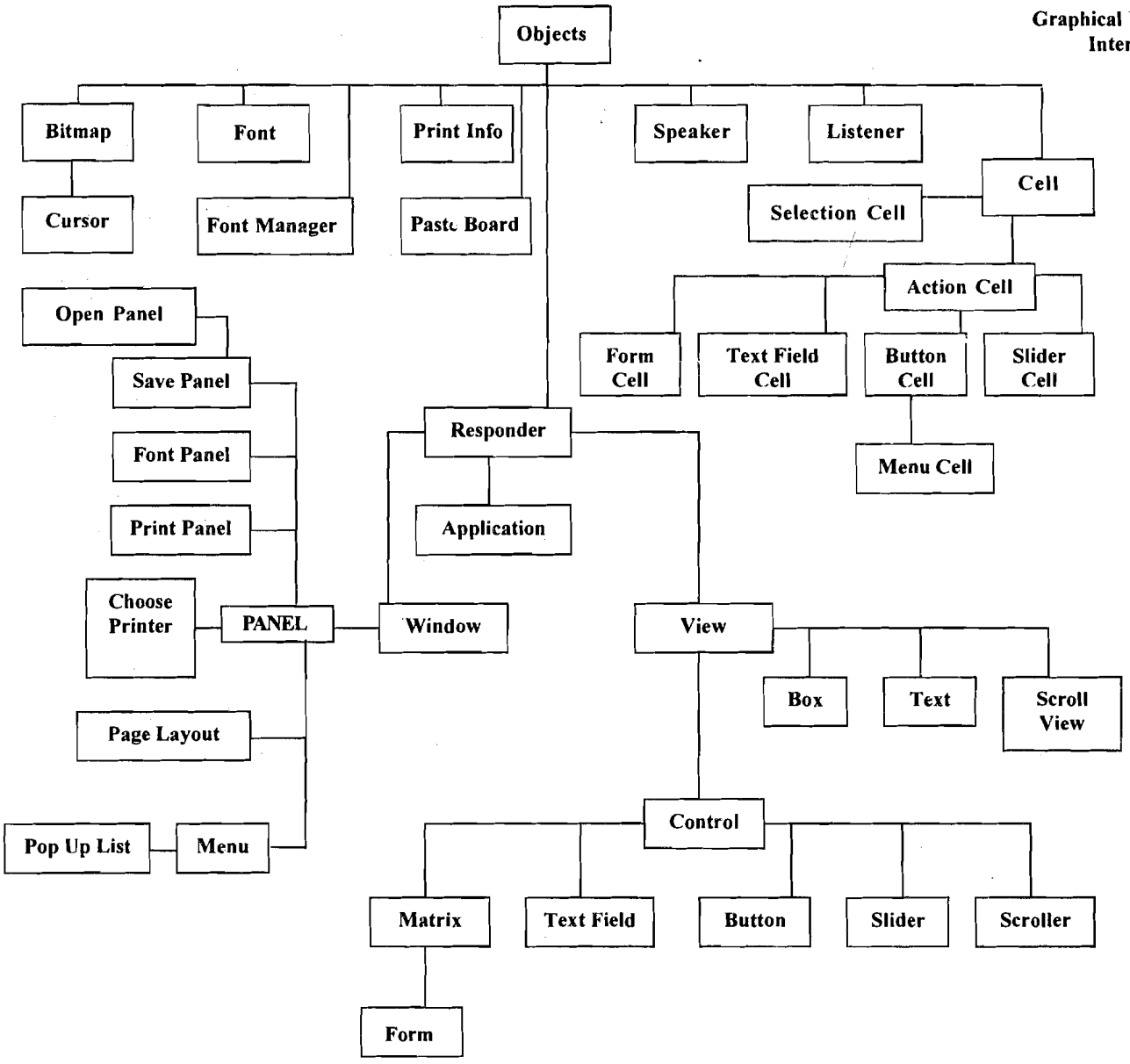


Figure 16: Application Kit

Application Kit

The Application Kit provides an extensive library of predefined classes. The hierarchy of classes is shown in *Figure 16.* These classes provide functionality to define user interfaces composed of menus, windows, buttons, slide bars, and sound. Each class within the hierarchy defines the behaviour of its objects. For example, a menu knows how to handle mouse events, when clicking on a menu item. Window objects know how to resize the window.

The Application Kit can be extended by adding new subclasses to the hierarchy. To add new subclasses, the class definition is written using the Objective C language. Each NeXT machine comes with a copy of Objective C. Objective C is an object-oriented extension of C. The language is a superset of C incorporating object orientation features from **Smalltalk**. Just like Smalltalk, it comes with a large collection of predefined classes to simplify the software development task. The language supports abstract data types, inheritance, and operator overloading. Unlike C++, Objective C does not extend the definition of any existing C language construct. It relies totally on the introduction of new constructs and operates to perform tasks such as class definition or message passing.

To develop user interfaces, designers can use Application Kit from Objective C directly. This would be very much like developing an application using MacApp. They can create instances of objects from the Application Kit hierarchy and modify the

attributes by calling the methods attached to the class definition. But the other method of defining user interface, using the Interface Builder, is much easier than coding it entirely in Objective C.

Designing User Interfaces with Interface Builder

The Interface Builder provides an easy to use utility to design a user interface using the muse and the screen. The Interface Builder is similar to an icon editor or screen painter. Designers can define the screen layout by selecting the screen objects from the Interface Builder palettes. The Interface Builder also helps to define the user-defined class and make connections between objects. Not all of the coding is automatic; the Application Kit and the Objective C language are also needed.

Defining a user interface using the Interface Builder requires the following steps:

1) **Define layout of screen:** the interface designer defines a user interface by simply selecting screen objects from the Interface Builder palettes, see *Figure 17.* After picking an object from a palette using the mouse, the object can be dragged into the destination window and resized as desired.

   The Interface Builder palettes include objects such as windows, menus, buttons, fields radio buttons and more. At the top of the palettes window, three buttons allow the interface developer to select a wide array of user-interface objects.

2) **Define the user-defined classes:** the developer defines a new class definition using the Classes Window. The Classes Window allows the developer to extend the Application Kit class hierarchy. The developer navigates through the class hierarchy and creates a new subclass within the hierarchy. Methods and outlets (see next step) are defined for this new class definition. When a class is defined this way, only the template of the class is created.

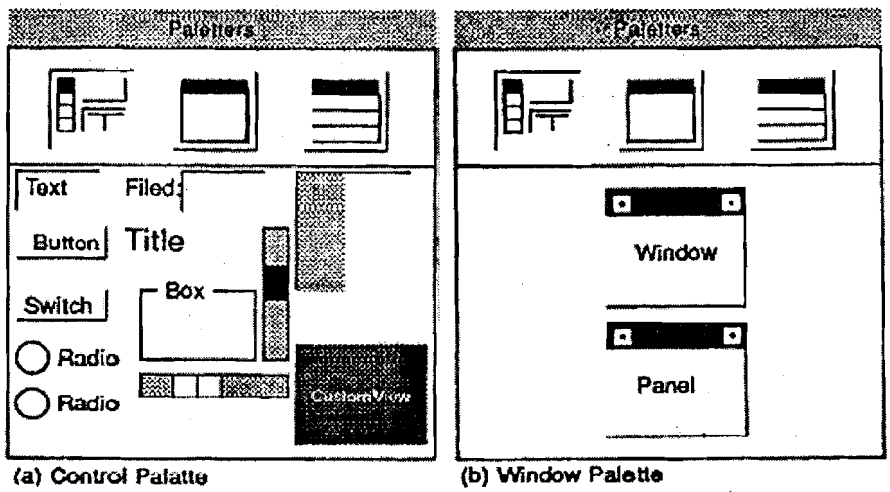


Figure 17: NeXT interface Builder palette

3) **Making connections:** up to this point we have defined the layout of the user interface. At this step, the developer needs to make connections among application objects. For example, when a scroller's slide bar is moved, a message is sent to an application object to perform certain actions like shifting the text. Again the Inspector is used to connect user interface objects with methods of classes within the application.

4) **Actual application code:** the previous steps are handled by Interface Builder directly. The last step is accomplished by writing the application code in Objective C. When the developer is done with the first two steps, the Interface Builder defines the source files necessary to build the application. These files contain the template for the class definitions and the connections made among objects. At this stage, the developer needs to extend these source files. Extensions are made to specify the logic of the program and the code for method definitions.

☞ **Check Your Progress 4**

1) What are the major components of NeXTSTEP? How do these elements

.....

.....

.....

2) How are applications written in NeXTSTEP environment?

.....

.....

.....

1.9 SUMMARY

The GUI has already made a tremendous contribution to the increased usability of computer systems. It is with great excitement that we look forward to future innovations in human-computer interfaces. GUI development is at the vanguard of creativity in many areas, such as ergonomics, software development tools, computer graphics and linguistics to name just a few. The past decade has seen a rapid change in the understanding and definition of GUIs, but we have only been through the infancy of GUIs. There remains much to be done in terms of increasing the productivity of computer users, standardising operations across different architectures and adapting the human-computer interface to non-traditional applications. In this unit we discussed several issues related to GUI including functioning of several popular packages.

1.10 SOLUTIONS/ANSWERS

☞ **Check Your Progress 1**

1) GUI is a system that allows creation and manipulation of user interfaces employing windows, menus, icons, dialog boxes, etc.

   The basic features of GUI are:

   - Pointing device such as mouse, which controls cursor.
   - Point and shoot functionality under control of device, which cause screen menus to appear or disappear.
   - Support of windows, which graphically display the status of a computer program.
   - Icons that represent files, directories and other application and system entities.
   - Support of graphical metaphors such as pull-down menus, dialog boxes, buttons, slides that let the programmer and user tell the computer what to do and how to do it.

2) a) When a screen is split into several independent regions each one is called a window. Several applications can display results simultaneously in different windows. The user can switch from one window to another window. Windowing systems have capabilities to display windows either tiled or lapped. Users can organize the screen by resizing the window or moving related windows closer.

b) No model answer is given.

c) No model answer is given.

d) No model answer is given.

3) Bit-mapped display is made up of tiny dots (pixel) that is independently addressable and has much finer resolutions than character displays. Bit-mapped displays have advantages over character displays. One of the major advantages is of graphic capability.

Check Your Progress 2

1) There are four major components of GUI.

i) A windowing system

ii) An emerging model

iii) An application program interface

iv) A set of tools and frameworks for creating interfaces and developing integrated application.

The API (Application Program Interface) is a set of programming language functions that allow the programmer to specify how the actual application will control the menus scroll bars and icons that appear on the screen. Like in windowing models, APIs align with particular GUIs. The features vary from package to package.

2) There are several ways MS-Windows enhance DOS environment.

i) Device independence. It presents a device independent interface to applications. Unlike most of today's DOS applications, a window application is not bound to the underlying hardware such as mouse, keyboard or display windows.

ii) Multitasking. Users can have several applications in progress at the same time. Each application can be active in a separate window.

iii) Memory management. Windows also provide memory management, the 640K limitation of MS-DOS. An application has the ability to extend memory and share data segments with other applications.

iv) Data sharing. Clipboards allow data transfer between application clipboard. Any type of data can be transferred from one window to another through the clipboard.

3) No model answer is given.

Check Your Progress 3

1) The toolbox provides a collection of utilities to access and manipulate Macintosh hardware and software resources. It provides a set of utilities to manipulate windows menu bar and dialog boxes. Some of the other utilities provided are:

i) Fonts manager: allows manipulation of system and user defined fonts.

ii) Event manager: provides monitoring of events generated by keyboard and keypad.

iii) Text edit: provides simple text editing capabilities such as cut and paste.

iv) Toolbox utilities: provides a set of general routines to manipulate strings, fixed point arithmetic and so forth.

2) No model answer is provided.

3) The X window system does not define any particular style of interface but provides a mechanism for supporting many styles. X is also network-based in contrast with other GUIs, or the X architecture is based on the premise that an application can run on one computer while the graphical presentation of the application's output and the responses from the user can occur on another computer.

Check Your Progress 4

1) No model answer is given.

2) No model answer is given.

1.11 FURTHER READINGS

1) *Communication of ACM*, April 1993.

2) *Object Orientation: Concepts, Languages, Databases. User Interfaces*, Khosafian, Setrag & Razmik Abnours, New York; Wiley & Sons, 1990.

UNIT 2 INTRODUCTION TO OPERATING SYSTEM

Structure **Page Nos.**

2.0 INTRODUCTION

An operating system is a system software which may be viewed as an organized collection of software consisting of procedures for operating a computer and providing an environment for execution of programs. It acts as an interface between users and the hardware of a computer system.

There are many important reasons for studying operating systems. Some of them are:

1) User interacts with the computer through the operating system in order to accomplish his task since it is his primary interface with a computer.

2) It helps users to understand the inner functions of a computer very closely.

3) Many concepts and techniques found in the operating system have general applicability in other applications.

The introductory concepts and principles of an operating system will be the main issues for discussion in this unit. The unit starts with the basic definition and then goes on to explain the stages of evolution of operating systems. It further gives details of several approaches to operating system design.

In the last two subsections of the unit we classify an advanced operating system and explain some characteristics of modern operating systems.

2) No model answer is provided.

3) The X window system does not define any particular style of interface but provides a mechanism for supporting many styles. X is also network-based in contrast with other GUIs, or the X architecture is based on the premise that an application can run on one computer while the graphical presentation of the application's output and the responses from the user can occur on another computer.

Check Your Progress 4

1) No model answer is given.

2) No model answer is given.

1.11 FURTHER READINGS

1) *Communication of ACM*, April 1993.

2) *Object Orientation: Concepts, Languages, Databases. User Interfaces*, Khosafian, Setrag & Razmik Abnours, New York; Wiley & Sons, 1990.

UNIT 2 INTRODUCTION TO OPERATING SYSTEM

Structure **Page Nos.**

2.0 INTRODUCTION

An operating system is a system software which may be viewed as an organized collection of software consisting of procedures for operating a computer and providing an environment for execution of programs. It acts as an interface between users and the hardware of a computer system.

There are many important reasons for studying operating systems. Some of them are:

1) User interacts with the computer through the operating system in order to accomplish his task since it is his primary interface with a computer.

2) It helps users to understand the inner functions of a computer very closely.

3) Many concepts and techniques found in the operating system have general applicability in other applications.

The introductory concepts and principles of an operating system will be the main issues for discussion in this unit. The unit starts with the basic definition and then goes on to explain the stages of evolution of operating systems. It further gives details of several approaches to operating system design.

In the last two subsections of the unit we classify an advanced operating system and explain some characteristics of modern operating systems.

2.1 OBJECTIVES

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

- List stages of evolution of operating systems:
- Classify different types of operating systems, and
- Compare different approaches to operating system design.

2.2 WHAT IS AN OPERATING SYSTEM?

An operating system is an essential component of a computer system. The primary objectives of an operating system are to make the computer system convenient to use and to utilise computer hardware in an efficient manner.

An operating system is a large collection of software, which manages the resources of the computer system, such as **memory, processor, file system** and **input/output devices.** It keeps track of the status of each resource and decides who will have control over computer resources, for how long and when. The positioning of an operating system in the overall computer system is shown in *Figure 1.*

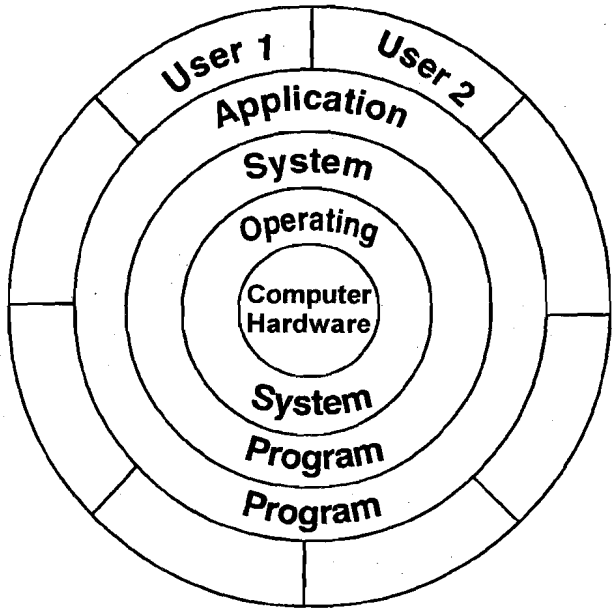


Figure 1: Component of Computer System

From the Figure, it is clear that the operating system directly controls computer hardware resources. Other programs rely on facilities provided by the operating system to gain access to computer system resources. There are two ways one can interact with the operating system:

1) By means of operating **System Call** in a program

2) Directly by means of **Operating System Commands.**

System Call: System calls provide the interface to a running program and the operating system. The user program receives operating system services through the set of system calls. Earlier these calls were available in assembly language instructions but now a day these features are supported through high level languages like C, Pascal

etc., which replace assembly language for system programming. The use of system calls in C or Pascal programs very much resemble pre-defined functions or subroutine calls.

As an example of how system calls are used, let us consider a simple program to copy data from one file to another. In an interactive system, the following system calls will be generated by the operating system:

- Prompt message for inputting two file names and reading it from the terminal.
- Open source and destination file.
- Prompt error message in case the source file cannot be open because it is protected against access or because the destination file cannot be created since there is already a file with this name.
- Read the source file
- Write into the destination file
- Display status information regarding various Read/Write error conditions. For example, the program may find that the end of the file has been reached or that there was a hardware failure. The write operation may encounter various errors, depending upon the output device (no more disk space, physical end of tape, printer out of paper and so on).
- Close both files after the entire file is copied.

As we can observe, a user program makes heavy use of the operating system. All interaction between the program and its environment must occur as the result of requests from the program to the operating system.

Operating System Commands: Apart from system calls, users may interact with the operating system directly by means of operating system commands.

For example, if you want to list files or sub-directories in **MS-DOS**, you invoke **dir** command. In either case, the operating system acts as an interface between users and the hardware of the computer system. The fundamental goal of computer systems is to solve user problems. Towards this goal computer hardware is designed. Since the bare hardware alone is not very easy to use, programs (software) are developed. These programs require certain common operations, such as controlling peripheral devices. The command function of controlling and allocating resources are then brought together into one piece of software; the operating system.

To see what operating systems are and what operating systems do, let us consider how they have evolved over the years. By tracing their evolution, we can identify the common elements of operating systems and examine how and why they have developed as they have.

2.3 EVOLUTION OF OPERATING SYSTEMS

An operating system may process its task serially (sequentially) or concurrently (several tasks simultaneously). It means that the resources of the computer system may be dedicated to a single program until its completion or they may be allocated among several programs in different stages of execution. The feature of the operating system to execute multiple programs interleaved fashion or different time cycles is called **multiprogramming systems.** In this section, we will try to trace the evolution of the operating system. In particular, we will describe **serial processing, batch processing** and **multiprogramming.**

2.3.1 Serial Processing

Programming in 1's and 0's (machine language) was quite common for early computer systems. Instructions and data used to be fed into the computer by means of console switches or perhaps through a hexadecimal keyboard. Programs used to be started by loading the program computer register with the address of the first instruction of a program and its result (program) used to be examined by the contents of various registers and memory locations of the machine. Therefore, programming in this style caused a low utilisation of both users and machine.

The advent of Input/output devices, such as punched cards, paper tape and language translators (Compiler /Assemblers) brought a significant step in computer system utilisation. Programs started being coded into programming language first changed into object code (binary code) by translator and then automatically loaded into memory by a program called **loader.** After transferring control to the loaded program, the execution of a program begins and its result gets displayed or printed. Once in memory, the program may be re-run with a different set of input data.

The process of development and preparation of a program in such an environment is slow and cumbersome due to serial processing and numerous manual processing. In a typical sequence first the editor is called to create a source code of user program written in programming language, then the translator is called to convert a source code into binary code and then finally the loader is called to load the executable program into the main memory for execution. If syntax errors are detected, the whole process must be redone from the beginning.

The next development was the replacement of card-decks with standard input/output and some useful library programs, which were further linked with user programs through system software called **linker.** While there was a definite improvement over machine language approach, the serial mode of operation is obviously not very efficient. This results in low utilization of resources.

2.3.2 Batch Processing

Utilisation of computer resources and improvement in programmer's productivity was still a major problem. During the time that tapes were being mounted or the programmer was operating the console, the CPU was sitting idle.

The next logical step in the evolution of the operating system was to automate the sequencing of operations involved in program execution and in the mechanical aspects of program development. Jobs with similar requirements were batched together and run through the computer as a group. For example, suppose the operator received one FORTRAN program, one COBOL program and another FORTRAN program. If he runs them in that order, he would have to set up for FORTRAN program environment (loading the FORTRAN compiler tapes), then set up COBOL program and finally FORTRAN program again. If he runs the two FORTRAN programs as a batch, however, he could set up only once for FORTRAN thus saving operator's time.

Batching similar jobs brought utilisation of system resources quite a bit. But there were still problems. For example, when a job is stopped, the operator would have to notice that fact by observing the console, determine why the program stopped and then load the card reader or paper tape reader with the next job and restart the computer. During this transition from one job to the next, the CPU sat idle.

To overcome this idle time, a small program called a **resident monitor** was created which is always resident in the memory. It automatically sequenced one job to another job. The resident monitor acts according to the directives given by a programmer through **control cards, which** contain information like marking of job's beginnings and endings, commands for loading and executing programs, etc. These commands belong to **job control language**. These job control language commands are included with user program and data. Here is an example of job control language commands.

$COB	- Execute the COBOL compiler
$JOB	- First card of a job
$END	- Last card of a job
$LOAD	- Load program into memory
$RUN	- Execute the user program

Figure 2 shows a sample card deck set up for a simple batch system.

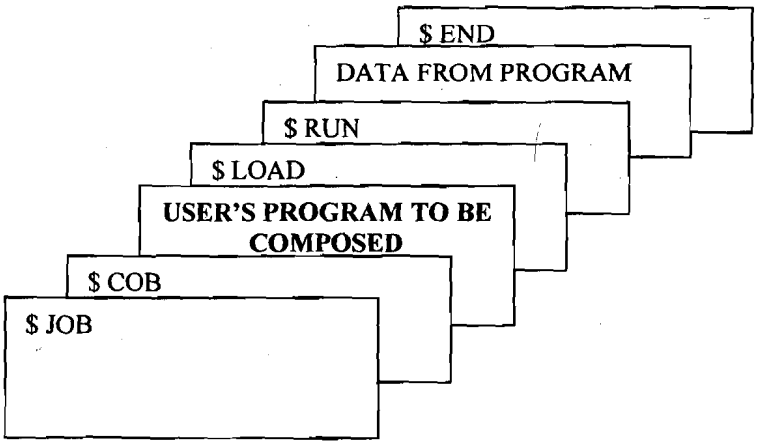


Figure 2: Card Deck for Cobol Program for a Simple Batch System

With sequencing of program execution mostly automated by batch operating system, the speed discrepancy between the fast CPU and the comparatively slow input/output devices such as card readers, printers emerged as a major performance bottleneck. Even a slow CPU works in the microsecond range, with millions of instructions per second. A fast card reader, on the other hand, might read 1200 cards per minute. Thus, the difference in speed between the CPU and its input/output devices may be three orders of magnitude or more.

The relative slowness of input/output devices can mean that the CPU is often waiting for input/output. As an example, an assembler or compiler may be able to process 300 or more cards per second. A fast card reader, on the other hand, may be able to read only 1200 cards per minute. This means that assembling or compiling a 1200 card program would require only 4 seconds of CPU time but 60 seconds to read. Thus, the CPU would be idle for 56 out of 60 seconds or 93.3 per cent of the time. The resulting CPU utilization is only 6.7 per cent. The process is similar for output operations. The problem is that while an input/output is occurring, the CPU is idle, waiting for the input/output to complete; while the CPU is executing, input/output devices are idle.

Over the years, of course, improvements in technology resulted in faster input/output devices. But CPU speed increased even faster. According to Moore's Law, CPU speed is getting doubled every 18 months. Therefore, the need was to increase the throughput and resources utilization by overlapping input/output and processing operations. Channels, peripheral controllers and later dedicated input/output processors brought a major improvement in this direction. DMA (Direct Memory Access) chip which directly transfers the entire block of data from its own buffer to main memory without intervention by the CPU was a major development. While the CPU is executing, DMA can transfer data between high-speed input/output devices and the main memory. The CPU requires to be interpreted per block only by DMA. Apart from DMA, there are two other approaches to improving system performance by overlapping input, output and processing. These are called **buffering** and **spooling**.

Buffering is a method of overlapping input, output and processing of a single job. The idea is quite simple. After data has been read and the CPU is about to start operating on it, the input device is instructed to begin the next input immediately. The CPU and input device are then both busy. With luck, by the time the CPU is ready for the next data item, the input device will have finished reading it. The CPU can then begin processing the newly read data, while the input device starts to read the following data. Similarly, this can be done for output. In this case, the CPU creates data that is put into a buffer until an output device can accept it.

If the CPU is, on the average, much faster than an input device, buffering will be of little use. If the CPU is always faster, then it always finds an empty buffer and has to wait for the input device. For output, the CPU can proceed at full speed until, eventually, all system buffers are full. Then the CPU must wait for the output device. This situation occurs with **input/output bound** jobs where the amount of input/output device, the speed of execution is controlled by the input/output device, not by the speed of the CPU.

A more sophisticated form of input/output buffering is called **SPOOLING** (Simultaneous Pheripheral Operation On Line) which essentially uses the hard disk (secondary memory) as a very large buffer (*Figure 3*) for reading and for storing output files.

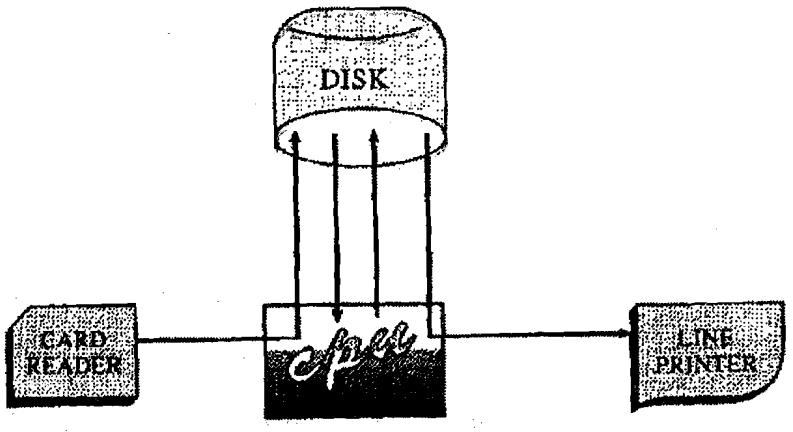


Figure 3: Spooling

Buffering overlaps input, output and processing of a single job **whereas Spooling allows CPU to overlap the input of one job with computation and output of other jobs.** Therefore this approach is better than buffering. Even in a simple system, the spooler may be reading the input of one job while printing the output of a different job.

2.3.3 Multiprogramming

Buffering and Spooling improve system performance by overlapping the input, output and computation of a single job, but both of them have their limitations. A single user cannot always keep CPU or I/O devices busy at all times. Multiprogramming offers a more efficient approach to increase system performance. In order to increase the resource utilisation, systems supporting multiprogramming approach allow more than one job (program) to reside in the memory to utilise CPU time at any moment. More number of programs competing for system resources better will mean better resource utilisation.

The idea is implemented as follows. The main memory of a system contains more than one program *(Figure 4)*.

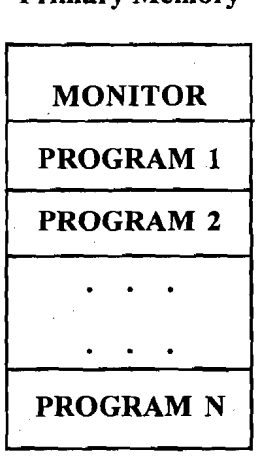
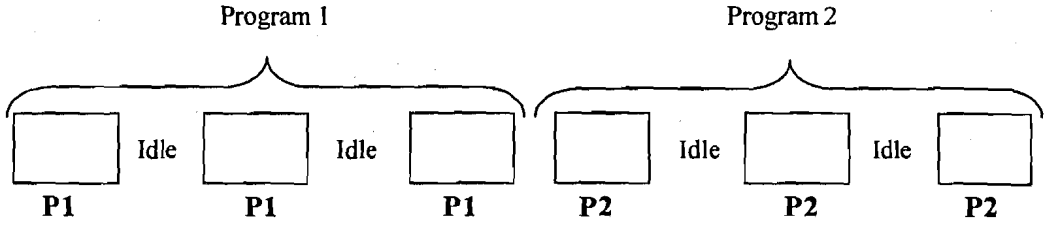
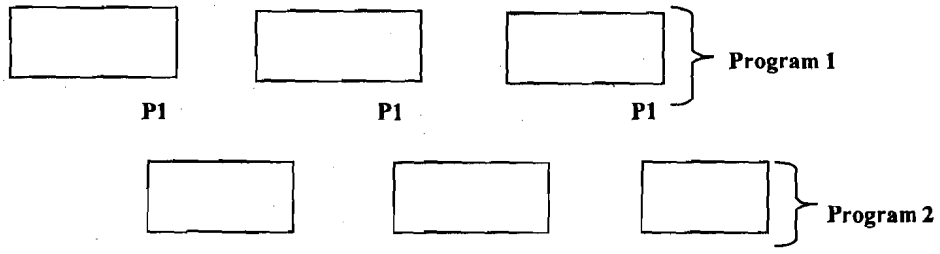


Figure 4: Memory Layout in Multiprogramming Environment

The operating system picks one of the programs and starts executing. During execution of program 1 it needs some I/O operation to complete in a sequential execution environment (*Figure 5a*). The CPU would then sit idle whereas in a multiprogramming system, (*Figure 5b*) the operating system will simply switch over to the next program (program 2).



(a): Sequential Execution



(b): Execution in Multiprogramming Environment

Figure 5: Multiprogramming

When that program needs to wait for some I/O operation, it switches over to program 3 and so on. If there is no other new program left in the main memory, the CPU will pass its control back to the previous programs.

Multiprogramming has traditionally been employed to increase the resources utilisation of a computer system and to support multiple simultaneously interactive users (terminals).

Compared to the operating system, which supports only sequential execution, the multiprogramming system requires some form of CPU and memory management strategies, which will be discussed in the next section.

2.4 OPERATING SYSTEM STRUCTURE

Since operating system is a very large and complex software, supports a large number of functions. It should be developed as a collection of several smaller modules with carefully defined inputs, outputs and functions rather than a single piece of software. In this section, we will examine different operating system structure.

2.4.1 Layered Structure Approach

The operating system architecture based on layered approach consists of a number of layers (levels), each built on top of lower layers. The bottom layer is the hardware; the highest layer is the user interface. The first system constructed in this way was the THE system built by E.W. Dijkestra (1968) and his students. The **THE** system was a simple batch operating system which had 32k of 27 bit words.

The system supported 6 layers (*Figure 6*).

5	User Programs
4	Buffering for I/O Devices
3	Device Driver
2	Memory Manager
1	CPU Scheduling
0	Hardware

Figure 6: The layered structure of THE operating system

As shown in *Figure 6*, layer 0 dealt with hardware; the higher layer 1 handled allocation of jobs to processor. The next layer implemented memory management. Level 3 contained the device driver for the operator's console. By placing it, as well as I/O buffering, at level 4, above memory management, the device buffers could be placed in virtual memory. The I/O buffering was also above the operator's console, so that I/O error conditions could be output to the operator's console.

The main advantages of the layered approach is modularity which helps in debugging and verification of the system easily. The layers are designed in such a way that it uses operation and services only of a layer below it. A higher layer need not know how these operations are implemented, only what these operations do. Hence each layer hides implementation details from higher-level layers. Any layer can be debugged without any concern about the rest of the layer.

The major difficulty with the layered approach is definition of a new level i.e. how to differentiate one level from another. Since a layer can use the services of a layer below it, it should be designed carefully. For example, the device driver for secondary memory must be at a lower level than the memory management routines since memory management requires the ability to use the backing store.

2.4.2 Virtual Machine

It is a concept which creates the illusion of a real machine. It is created by a virtual machine operating system that makes a single real machine appear to be several real machines. This type of situation is analogous to the communication line of Telephone Company, which enables separate and isolated conversations over the same wire(s).

The following *Figure* illustrates this concept.

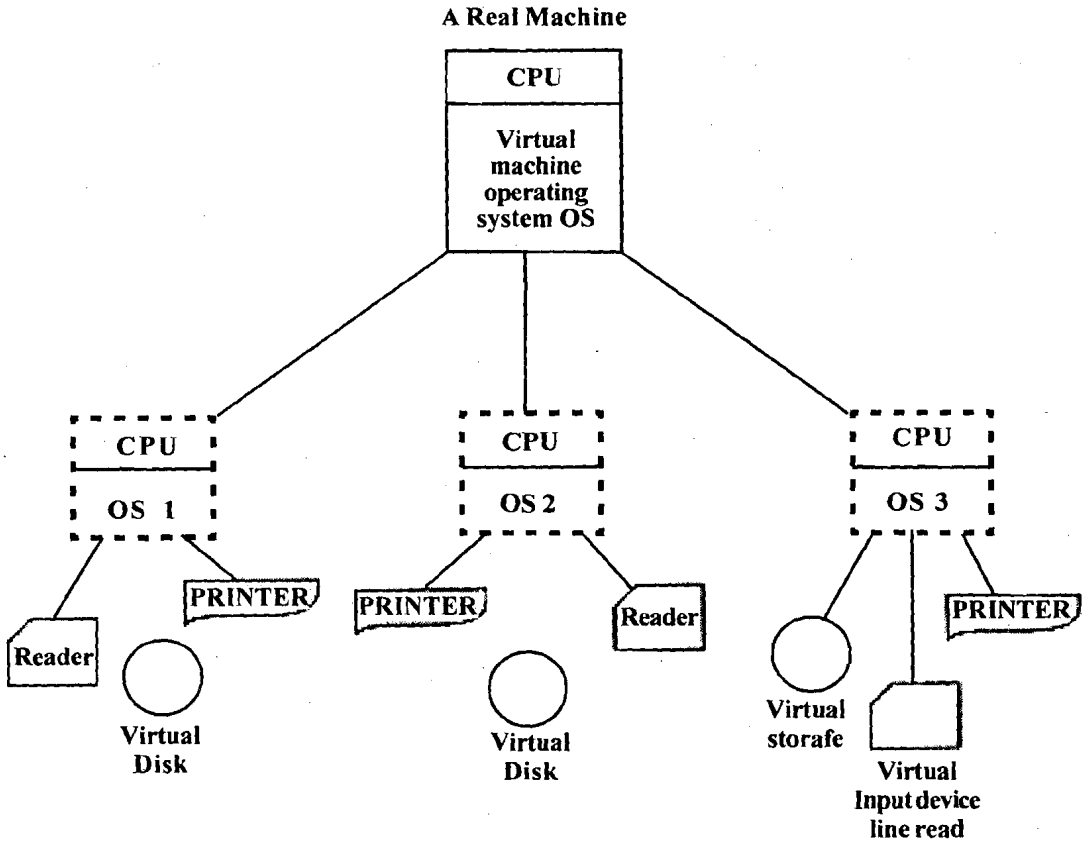


Figure 7: Creation of several virtual machines by a single physical machine

From the user's viewpoint, a virtual machine can be made to appear very similar to an existing real machine or they can be entirely different. An important aspect of this technique is that each user can run the operating system of his own choice. This fact is depicted by OS_1 (Operating System 1), OS_2, OS_3 etc. as in *Figure 7*.

To understand this concept, let us try to understand the difference between conventional multiprogramming system (Figure 8) and virtual machine multiprogramming (*Figure 9*). In conventional multiprogramming, processes are allocated a portion of the real machine resources. The same machine resources are distributed among several processes.

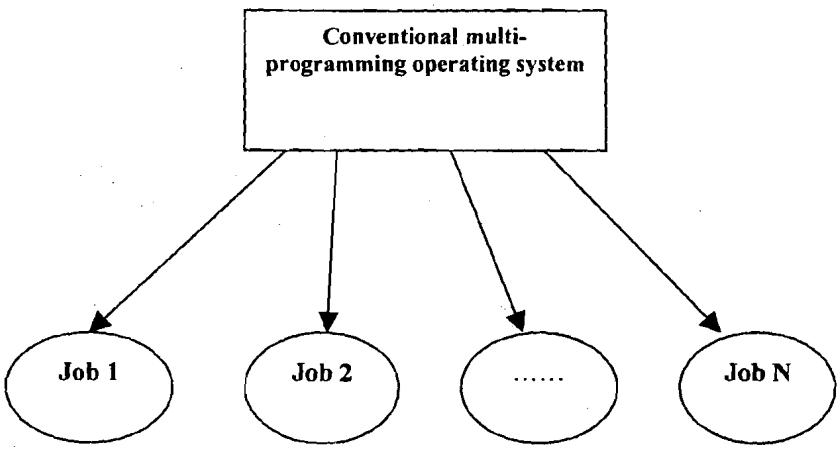


Figure 8: Conventional multiprogramming

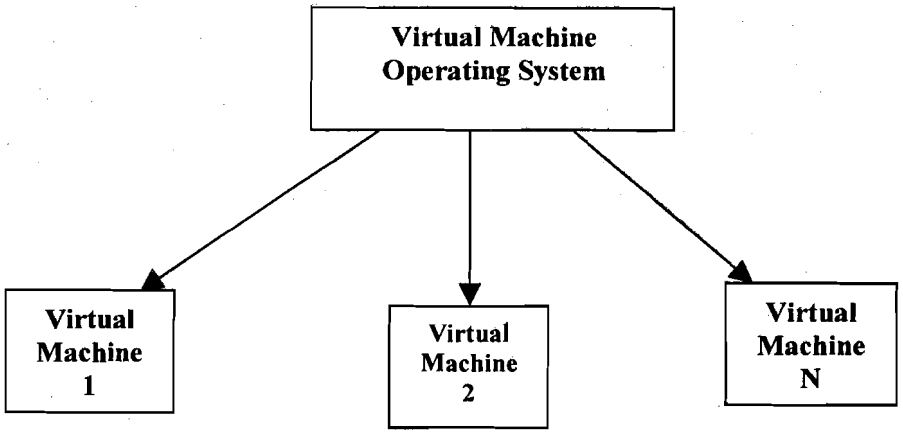


Figure 9: Virtual Machine Multiprogramming

In the virtual multiprogramming system, a single real machine gives the illusion of several virtual machines, each having its own virtual processor, storage and I/O devices possibly with much larger capacities.

The virtual machine has many uses and advantages:

1) Concurrent running of dissimilar operating systems by **different users.**

2) **Elimination of certain** conversion problems.

3) **Software development:** Programs can be developed and debugged for machine configurations that is different from those of host (for example virtual operating system VM/370 can produce virtual 370 that are different from the real 370 such as larger main memory).

4) **Security and Privacy:** The high degree of separation between independent virtual machines aids in ensuring privacy and security. The most widely used operating system in this category is VM/370. It manages IBM/370 computer and creates the illusion that each of several users has a complete system 370 (including wide range of I/O devices) which can run different operating systems at once, each of them on its own virtual machine.

It is also possible through software to share files existing on physical disk and much information through virtual communication software.

The virtual machines are created by sharing the resources of the physical computer. CPU scheduling can be used to share the CPU and make it appear that users have their own processor. Users are thus given their own virtual machine. They can run on their virtual machines any software desired. The virtual machine software is concerned with multiprogramming multiple virtual machines onto a physical machine but need not consider any other software support from user.

The heart of the system, known as the **virtual machine monitor,** runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up, as shown in *Figure 7*. However, unlike all other operating systems, these virtual machines are not extended machines, with files and other nice features. Instead, they are exact copies of the bare hardware, including kernel/user mode, I/O, interrupts, and everything else the real machine has.

Because each virtual machine is identical to the true hardware, each one can run any operating system that will run directly on the hardware. In fact, different virtual machines can, and usually do, run different operating systems. Some run one of the descendants of OS/360 for batch processing, while other ones run a simple, single-user, interactive system called CMS (Conversational Monitor System) for time-sharing users.

When a CMS program executes a system call, the call is trapped to the operating system in its own virtual machine, not to VM/370; just as it would if it were running on a real machine instead of a virtual one. CMS then issues the normal hardware I/O instructions for reading its virtual disk or whatever is needed to carry out the call. These I/O instructions are trapped by VM/370, which then performs them as part of its simulation of the real hardware. By making a complete separation of the functions of multiprogramming and providing an extended machine, each of the pieces can be mush simpler and more flexible.

The virtual machine concept has several advantages. Notice that there is complete protection. Each machine is completely isolated from all other virtual machines, so there is no problem with protection. On the other hand, there is no sharing. To provide sharing, two approaches have been implemented. First, it is possible to share a minidisk. This scheme is modeled after a physical shared disk, but implemented by software. With this technique, files can be shared. Second, it is possible to define a network of virtual machines, each of which can send information over the virtual communications network. Again, the network is modeled after physical communication networks, but implemented in software.

Such a virtual machine system is a perfect vehicle for operating systems research and development. Normally changing an operating system is a difficult process. Since operating systems are large and complex programs, it is difficult to be sure that a change in one point does not cause obscure bugs in some other part. This situation can be particularly dangerous because of the power of the operating system. Since the operating system executes in monitor mode, a wrong change in a pointer could cause an error that would destroy the entire file system. Thus, it is necessary to test all changes to the operating system carefully.

But the operating system runs on and controls the entire machine. Therefore, the current system must be stopped and taken out of use, while changes are made and tested. This is commonly called system development time. Since it makes the system unavailable to users, system development time is often scheduled late at night or on weekends.

A virtual machine system can eliminate much of this problem. System programmers are given their own virtual machine and system development is done on the virtual machine, instead of on a physical machine. The normal system operation seldom need be disrupted for system development.

2.4.3 Client-Server Model

VM/370 gains much in simplicity by moving a large part of the traditional operating system code (implementing the extended machine) into a higher layer itself. VM/370 is still a complex program because stimulating a number of virtual 370s is not that simple (especially if you want to do it efficiently).

A trend in modern operating systems is to take this idea of moving up into a higher layers even further, and remove as much as possible from the operating system, leaving a minimal **kernel**. The usual approach is to implement most of the operating system functions in user processes. To request a service, such as reading a block of a file, a user process (now known as the **client process**) sends the request to a **server process**, which then does the work and sends back the answer.

In this model, shown in *Figure 10,* all that the kernel does is to handle the communication between clients and servers. By splitting the operating system up into parts, each part is made to handle one facet of the system, such as file service, process service, and terminal service or memory service. This way, each part becomes small and manageable. Furthermore, because all the servers run as user-

mode processes, and not in kernel mode, they do not have direct access to the hardware. As a consequence, if a bug in the file server is triggered, the file service may crash, but this will not usually bring the whole machine down.

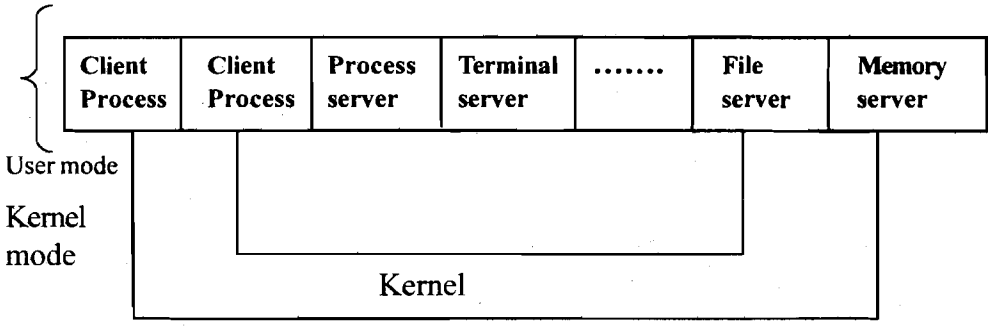


Figure 10: The Client-Server Model

Another advantage of the client-server model is its adaptability to use in distributed systems (*Figure 11*). If a client communicates with a server by sending it messages, the client need not know whether the message is handled locally in its own machine, or whether it was sent across a network to a server on a remote machine. As far as the client is concerned, the same thing happens in both cases: a request was sent and a reply came back.

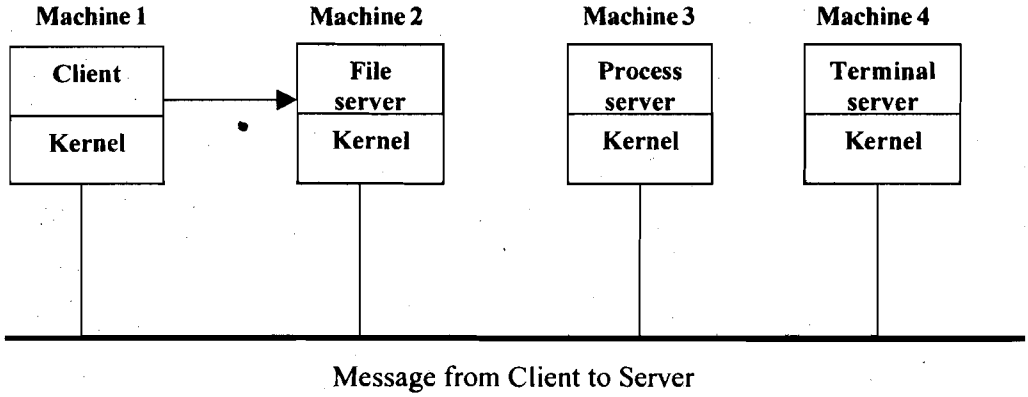


Figure 11: The Client-Server Model in a distributed System

The picture painted above of a kernel that handles only the transport of messages from clients to servers and back is not completely realistic. Some operating system functions (such as loading commands into the physical I/O device registers) are difficult, if not impossible, to do from user-space programs. There are two ways of dealing with this problem. One way is to have some critical server processes (e.g., I/O device drivers) actually run in kernel mode, with complete access to all the hardware, but still communicate with other processes using the normal message mechanism.

The other way is to build a minimal amount of **mechanism** into the kernel, but leave the **policy** decisions up to servers in user space. For example, the kernel might recognize that a message sent to a certain special address means to take the contents of that message and load it into the I/O device registers from some disk, to start a disk read. In this example, the kernel would not even inspect the bytes in the message to see if they were valid or meaningful; it would just blindly copy them into the disk's device registers. (Obviously some scheme for limiting such messages to authorized processes only must be used). The split between mechanism and policy is an important concept; it occurs again and again in operating systems in various contexts.

2.4.4 Kernel Approach

Kernel is that part of operating system which directly makes interface with hardware system. Its main functions are:

- To provide a mechanism for creation and deletion of processes.
- To provide processor scheduling, memory management and I/O management
- To provide a mechanism for synchronisation of processes so that processes synchronise their actions.
- To provide a mechanism for interprocess communication.

The UNIX operating system is based on kernel approach (*Figure 12*). It consists of two separable parts: (i) Kernel (ii) System Programs.

As shown in the *Figure 12,* the kernel is between system programs and hardware. The kernel supports the file system, processor scheduling, memory management and other operating system functions through system calls. The UNIX operating system supports a large number of system calls for process management and other operating system functions. Through these system calls, the program utilises the services of the operating system (kernel).

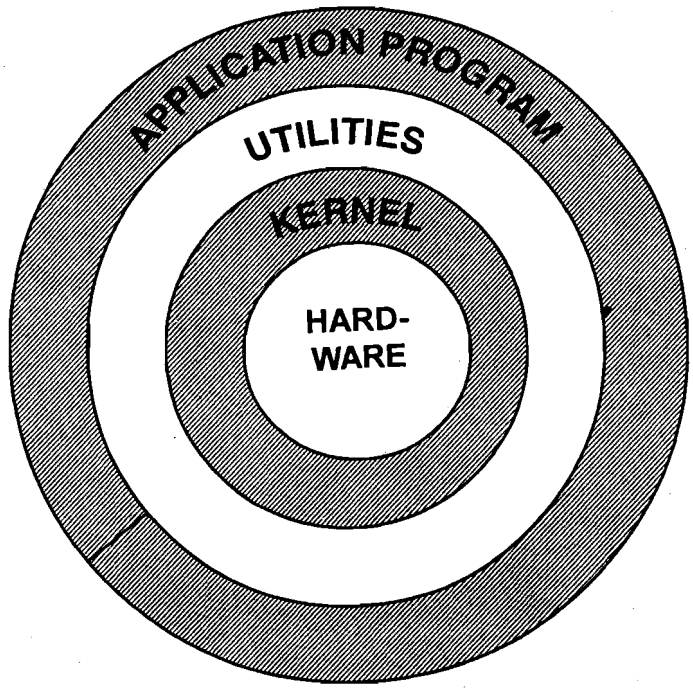


Figure 12: UNIX operating system structure

Check Your Progress 1

1) What is a system call?

.....

.....

.....

.....

2) Define the essential difference between:

   a) Spooling

   b) Buffering

2.5 CLASSIFICATION OF ADVANCED OPERATING SYSTEMS

The drive for advanced operating systems has come from two directions. First, it has come from advances in the architecture of multicomputer systems and is now driven by a wide variety of high-speed architectures. Hardware design of extremely fast parallel and distributed systems is fairly well understood. These architectures offer great potential for speed up but they also present a substantial challenge to operating system designers. The following *Figure 13* gives a broad classification of the advanced operating system.

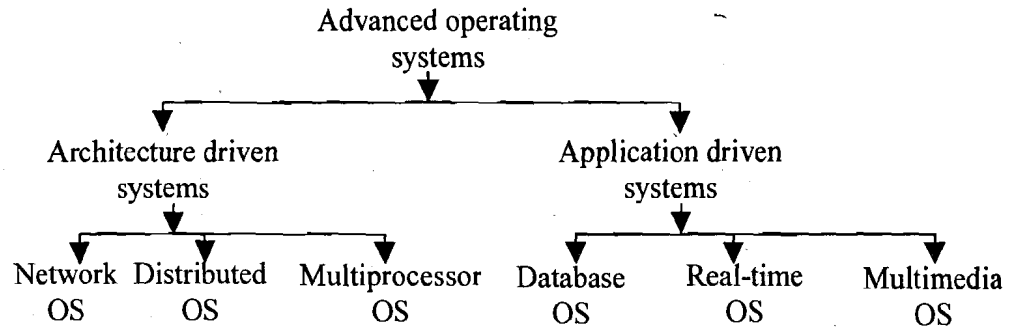


Figure 13: Classification of Advanced OS

A second class advanced operating system is driven by applications. There are several important applications that require special operating system support, as a requirement as well as for efficiency. General-purpose operating systems are too broad in nature and inefficient and fail to provide adequate support for such applications. Three specific applications, namely, multimedia operating systems, database systems and real-time systems, have received considerable attention in the past and the operating system issues for these systems have been extensively examined.

2.5.1 Architecture Driven Operating System

A brief discussion of three operating systems in the category of the Architecture Driven OS follows:

Network Operating System

A network operating system is a collection of software and associated protocols that allows a set of autonomous computers, which are interconnected by a computer network, to be used together in a convenient and cost-effective manner. In a network operating system, the users are aware of the existence of multiple computers and can log in to remote machines and copy files from one machine to an other machine.

Some of typical characteristics of network operating systems which make it different from a distributed operating system (discussed in the next section) are the followings:

- Each computer has its own private operating system instead of running part of a global system wide operating system.
- Each user normally works on his/her own system: using a different system requires some kind of remote login, instead of having the operating system dynamically allocate processes to CPUs.
- Users are typically aware of where each of their files are kept and must move file from one system to another with explicit file transfer commands instead of having file placement managed by the operating system.

The system has little or no fault tolerance; if 5% of the personal computers crash, only 5% of the users are out of business.

The network operating system offers many capabilities including:

- Allowing users to access the various resources of the network hosts.
- Controlling access so that only users in the proper authorization are allowed to access particular resources.
- Making the use of remote resources appear to be identical to the use of local resources.
- Providing up-to-the minute network documentation on-line.

As we said earlier, the key issue that distinguishes a network operating system from a distributed one is how aware the users are of the fact that multiple machines are being used. The visibility occurs in three primary areas; file system, protection and program execution.

Distributed Operating System

Distributed operating systems are operating systems for a network of autonomous computers connected by a communication network. A distributed operating system controls and manages the hardware and software resources of a distributed system such that its users view the entire system as a powerful monolithic computer system. When a program is executed in a distributed system, the user is not aware of where the program is executed or of the location of the resources accessed.

The basic issues in the design of a distributed operating system are the same as in a traditional operating system, viz., process synchronisation, deadlocks, scheduling, file systems, interprocess communication, memory and buffer management, failure recovery, etc. However, several idiosyncrasies of a distributed system, namely, the lack of both shared memory and a physical global clock, and unpredictable communication delays make the design of distributed operating systems much more difficult.

Network operating systems focus on the use of remote services and resources existing on a network of computer systems. Distributed operating systems focus on effective utilization of resources in distributed computing environments.

Distributed systems provide many advantages concerning cost-effectiveness of both computations and resources. The primary advantages are:

- Resource sharing
- Reliability
- Communication
- Incremental growth

Resource sharing has been the main motivation for distributed systems. The earliest form of a distributed system was a computer network which enabled the use of specialized hardware and software resources by geographically distant users. Resource sharing continues to be an important aspect of distributed systems today. However, the nature of distribution and sharing of resources has changed due to advances in networking technology. Sharing of resources is now equally meaningful in a local area network (LAN), for example sharing laser printers in the lab.

One aspect of reliability is availability of a resource despite failures in a system. A distributed environment can offer enhanced availability of resources through redundancy of resources and communication paths. For example, availability of a disk resource can be increased by having two or more disks located at different sites in the system. If one disk is unavailable due to a disk or site failure, a program can use some other disk. The availability of a data resource, e.g., a file, can be similarly enhanced by keeping copies of the file at various sites in the system.

Communication between users at different locations is greatly facilitated using a distributed system. There are two important aspects to communication. First, users have unique id's in a distributed system. Their use in communication automatically invokes the security mechanisms of the OS, thereby ensuring privacy and authenticity of communication. Second, use of a distributed system also implies continued availability of communication when users migrate to different sites of a system.

Distributed systems are capable of incremental growth, i.e., the capabilities of a system (e.g. its processing power) can be enhanced at a price proportional to the nature and size of the enhancement. A major advantage of this feature is that enhancements need not be planned in advance. This is in contrast to the classical mainframe architectures where enhancements often took the form of upgradation, that is, replacement of subsystems by more powerful ones—hence enhancement costs tended to be disproportionate to the nature and size of an enhancement.

Distributed systems today cover a wide spectrum of computer hardware, software and topological configurations; resource sharing services range from off-line access to real-time access and topologies vary from locally distributed to geographically distributed.

Multiprocessor Operating Systems

A typical multiprocessor system consists of a set of processors that share a set of physical memory blocks over an interconnection network. Thus, a multiprocessor system is a tightly coupled system where processors share an address space. A multiprocessor operating system controls and manages the hardware and software resources such that users view the entire systems as a powerful uniprocessor system; a user is not aware of the presence of multiple processors and the interconnection network.

The basic issues in the design of a multiprocessor operating system are the same as in a traditional operating system. However, the issues of process synchronisation, task scheduling, memory management, and protection and security become more complex because the main memory is shared by many physical processors.

2.5.2 Application Driven Operating System

A brief discussion of three operating systems in the category of the Architecture Driven OS follows:

Multimedia Operating System

The operating system is the shield of the computer hardware against all software components. It provides a comfortable environment for the execution of programs, and it ensures effective utilisation of the computer hardware. The operating system offers various services related to the essential resources of a computer: CPU, main memory, storage and all input and output devices.

For the processing of audio and video, multimedia application demands that humans perceive these media in a natural, error-free way. These continuous media data

originate at sources like microphones, cameras and files. From these sources, the data are transferred to destinations like loudspeakers, video windows and files located at the same computer or at a remote station. On the way from source to sink, the digital data are processed by at least some type of move, copy or transmit operation. In this data manipulation process there are always many resources which are under the control of the operating system. The integration of discrete and continuous multimedia data demands additional services from many operating system components.

The major aspect in this context is real-time processing of continuous media data. Process management must take into account the timing requirements imposed by the handling of multimedia data. Appropriate scheduling methods should be applied. In contrast to the traditional real-time operating systems, multimedia operating systems also have to consider tasks without hard timing restrictions under the aspect of fairness.

To obey timing requirements, single components are conceived as resources that are reserved prior to execution. This concept of resource reservation has to cover all resources on a data path, i.e. all resources that deal with continuous media. It also may affect parts of the application that process continuous media data. In distributed systems, for example, resource management also comprises network capacity.

The communication and synchronization between single processes must meet the restrictions of real-time requirements and timing relations among different media. The main memory is available as a shared resource to single processes.

In multimedia systems, memory management has to provide access to data with a guaranteed timing delay and efficient data manipulation functions. For instance, physical data copy operations must be avoided due to their negative impact on performance; buffer management operations (such as are known from communication systems) should be used.

Database Operating System

Database systems place special requirements on operating systems. These requirements have their roots in the specific environment that database systems support. A database system must support: the concept of a transaction; operations to store, retrieve, and manipulate a large volume of data efficiently; primitives for concurrency control, and system failure recovery. To store temporary data and data retrieved from secondary storage, it must have a buffer management scheme.

Real-time Operating System

The main characteristics of the real-time systems is the correctness of the computation. This correctness does not apply to error-free computation, but also on the time in which the result is processed and produced. Therefore, the real-time systems must execute its critical workload in time to prevent failures. Examples of applications requiring support of real-time systems are process control, air traffic control, guidance of missile systems, etc.

Real-time systems also place special requirements on operating system, which have their roots in the specific application that the real-time system is supporting. A distinct feature of real-time systems is that jobs have completion deadlines. A job should be completed before its deadline to be of use (in soft real-time system) or to avert a disaster (in hard real-time systems). The major issue in the design of real-time operating systems is the scheduling of jobs in such a way that a maximum number of jobs satisfy their deadlines. Other issues include designing languages and primitives to effectively prepare and execute a job schedule.

☞ **Check Your Progress 2**

1) What are the basic design issues in the distributed operating systems.

.....

.....

.....

2) What is the basic difference between Network operating systems and Distributed operating systems.

.....

.....

.....

2.6 CHARACTERISTICS OF MODERN OPERATING SYSTEM

Over the years, there has been a gradual evolution of operating system structure and capabilities. However, in recent years a number of new design elements have been introduced into both new operating systems and new releases of existing operating systems that create a major change in the nature of operating systems. These modern operating systems respond to new developments in hardware and new applications. Among the key hardware drivers are multiprocessor machines, greatly increased machine speed, high speed network attachments, and increasing the size and variety of memory storage devices. In the application arena, multimedia applications, Internet and Web access, and client/server computing have influenced operating system design.

The rate of change in the demands on operating systems requires not just modifications and enhancements to existing architectures but new ways of organising the operating system. A wide range of different approaches and design elements has been tried in both expiremental and commercial operating systems, but much of the work fits into the following categories:

- Microkernel architecture
- Multithreading
- Symmetric multiprocessing

Most operating systems, until recently, featured a large **monolithic kernel.** Most of what is thought of as operating system functionality is provided in these large kernels, including scheduling, file system, networking, device drivers, memory management and more. Typically, a monolithic kernel is implemented as a single process with all elements sharing the same address space. A **microkernel architecture** assigns only a few essential functions to the kernel, including address spaces, interprocessor communication (IPC), and basic scheduling. Other OS services are provided by processes, sometimes called servers, that run in user mode and are treated like any other application by the microkernel. This approach decouples kernel and server development. Servers may be customised to specific application or environment requirements. The microkernel approach simplifies implementation, provides flexibility, and is well suited to a distributed environment. In essence, a microkernel interacts with local and remote server processes in the same way, facilitating construction of distributed systems.

2.6.1 Microkernel Architecture

A **microkernel architecture** assigns only a few essential functions to the kernel, including address spaces, interprocess communication (IPC), and basic scheduling. Other OS services are provided by processes, sometimes called servers, that run in user mode and are treated like any other application by the microkernel. This approach decouples kernel and server development. Servers may be customised to specific application or environment requirements. The microkernels approach simplifies implementation, provides flexibility, and is well suited to a distributed environment. In essence, a microkernel interacts with local and remote server processes in the same way, facilitating construction of distributed systems.

2.6.2 Multithreading

Multithreading is a technique in which a process executing an application is divided into threads that can run concurrently. We can make the following distinction:

- **Threads:** A dispatchable unit of work. It includes a processor context (which includes the program counter and stack pointer) and its own data area for a stack (to enable subroutine branching). A thread executes sequentially and is interruptable so that the processor can turn to another thread.
- **Process:** A collection of one or more threads and associated system resources (such as memory containing code and data, open files, and devices). This corresponds closely to the concept of a program in execution. By breaking a single application into multiple threads, the programmer has great control over the modularity of the application and the timing of application related events.

Multithreading is useful for applications that perform a number of essentially independent tasks that do not need to be serialised. An example is a database server that listens for and processes numerous client requests. With multiple threads running within the same process, switching back and forth among threads involves less processor overhead than a major process switch between different processes. Threads are also useful for structuring processes that are part of the OS kernel as described in subsequent chapters.

2.6.3 Symmetric Multiprocessing

Until recently, virtually all single-user personal computers and workstations contained a single general-purpose microprocessor. As demands for performance increase and as the cost of microprocessors continues to drop, vendors have introduced computers with multiple microprocessors. To achieve greater efficiency and reliability, one technique is to employ **symmetric multiprocessing (SMP),** a term that refers to a computer hardware architecture and also to the operating system behaviour that reflects that architecture. A symmetric multiprocessor can be defined as a standalone computer system with the following characteristics:

1) There are multiple processors.
2) These processors share the same main memory and I/O facilities, interconnected by a communications bus or other internal connection scheme.
3) All processors can perform the same functions (hence the term Symmetric).

The operating system of an SMP schedules processes or threads across all of the processors. SMP has a number of potential advantages over uniprocessor architecture, including the following:

- **Performance:** If the work to be done by a computer can be organised so that some portions of the work can be done in parallel, then a system with multiple

processors will yield greater performance than one with a single processor of the same type. With multiprogramming, only one process can execute at a time; meanwhile all other processes are waiting for the processor. With multiprocessing, more than one process can be running simultaneously, each on a different processor.

- **Availability:** In a symmetric multiprocessor, because all processors can perform the same functions, the failure of a single processor does not halt the machine. Instead, the system can continue to function at reduced performance.
- **Incremental growth:** A user can enhance the performance of a system by adding an additional processor.
- **Scaling:** Vendors can offer a range of products with different price and performance characteristics based on the number of processors configured in the system.

It is important to note that these are potential rather than guaranteed benefits. The operating system must provide tools and functions to exploit the parallelism in an SMP system.

Multithreading and SMP are often discussed together, but the two are independent facilities. Even on a uniprocessor machine, multithreading is useful for structuring applications and kernel processes. An SMP machine is useful for non-threaded processes, because several processes can run in parallel. However, the two facilities complement each other and can be used effectively together.

An attractive feature of an SMP is that the existence of multiple processors is transparent to the user. The operating system takes care of scheduling of threads or processes on individual processors and of synchronisation among processors. A different problem is to provide the appearance of a single system for a cluster of separate computers – a multicomputer system. In this case, we are dealing with a collection of entities (computers), each with its own main memory, secondary memory, and other I/O modules. A distributed operating system provides the illusion of a single main memory space and a single secondary memory space, plus other unified access facilities, such as a distributed file system. Although clusters are becoming increasingly popular, and there are many cluster products on the market, the state of the art for distributed operating systems lags that of uniprocessor and SMP operating systems.

The most recent innovation in operating system design is the use of object oriented technologies. Object oriented design lends discipline to the process of adding modular extensions to a small kernel. At the operating system level, an object-based structure enables programmers to customise an operating system without disrupting system integrity. Object orientation also eases the development of distributed tools and a full-blown distributed operating system.

2.7 SUMMARY

The operating system is an essential component of system software, which consists of procedures for managing computer resources. Initially computers were operated from the front console. System software such as Assemblers, Loaders and Compilers greatly helped in software development but also required substantial setup time. To reduce the setup time an operator was hired and similar jobs were batched together.

Batch systems allowed automatic job sequenting by a resident monitor and improved the overall utilisation of systems greatly. The computer no longer had to wait for human operations – but CPU utilisation was still low because of slow speed of I/O

devices compared to the CPU. A new concept buffering was developed to improve system performance by overlapping the input, output and computation of a single job. Spooling was another new concept in improving the CPU utilisation by overlapping input of one job with the computation and output of other jobs.

Operating systems are now almost always written in a higher-level language (C, PASCAL, etc.). UNIX was the first operating system developed in C language. This feature improves their implementation, maintenance and portability.

The operating system provides a number of services. At the lowest level, there are system calls, which allow a running program to make a request from the operating system directly. At a higher level, there is a command interpreter, which supports a mechanism for a user to issue a request without writing a program.

In this unit, we began with tracing the evolution of the operating system through serial processing, batch processing and multiprogramming. We also presented different operating system models: Layered structured, Kernel based, virtual machine system and client server model.

At the end we presented classification and characteristics of emerging operating systems.

2.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) System calls provide the interface between a running program and the operating system. User programs receives OS services through the set of system calls.

2) Both buffering and spooling are used to improve system performance. Buffering overlaps input, output and processing of a single job whereas spooling allows CPU to overlap the input of the job with computation and output of another jobs. Spooling is better than buffering.

Check Your Progress 2

1) The basic design issues of a distributed operating system are the same as in a conventional operating system, viz., process synchronisation, deadlocks, scheduling, file systems, interprocess communication, memory and buffer management, failure recovery, etc., but what makes its design complexity are the lack of shared memory and common physical global clock and unpredictable communication delays.

2) The basic difference between the two is that the network operating system focuses on the use of remote services and resources existing on a network operating system whereas the distributed operating system focuses on the effective utilisation of resources on distributed environment.

2.9 FURTHER READINGS

1) *Operating System Concepts* by Abraham Silberschatz and James L. Peterson, Addision Wesely.

2) *Operating Systems Design and Implementation* by Andrew S. Tanenbaum, Prentice Hall of India.

UNIT 3 INTRODUCTION TO NETWORKING CONCEPT

Structure **Page Nos.**

3.0 INTRODUCTION

A network can consist of two computers connected together on a desk or it can consist of many Local Area Networks (LANs) connected together to form a Wide Area Network (WAN) across a continent. In simple terms it is an interconnected set of some objects. For decades, we have been familiar with the Radio, Television, Railways, Banks and various other types of networks. In recent years, the computer network, a new form of network is becoming more and more visible in our day-to-day life. A computer network is an interconnected set of autonomous computers.

Autonomous means each of them can function independent of others, i.e., each computer has individual processors. Simply we can say each computer (terminal, node) should not be a dump terminal. The key is that two or more computers are connected together by a medium and are sharing resources. These resources can be files, printers, hard drives, CPU or data. By using a computer network, people can send and receive back information more quickly.

3.1 OBJECTIVES

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

- define what a network is?
- understand what is the need of a computer network and the applications of networks;
- list types of networks, topologies and mediums finally, and
- how devices are connected though repeater, bridges, router, Gateway.

3.2 WHY COMPUTER NETWORKS?

Computer Networks offer a number of advantages to individuals and organization. Some of these are:

a) **Communication medium**: It offers a powerful communication medium among a group of people widely separated on the earth.

b) **Resource Sharing**: Resources like files, printers, hard drives, or CPU can be shared through a computer network.

c) **Higher Reliability:** If one computer **is** down; its workload can be taken over by the other computer. So it offers higher reliability than a centralized computing environment.

d) **Higher flexibility**: A heterogeneous system can be connected in a computer network, by which users get better flexibility.

e) **Scalable:** Computers and other equipments can be gradually added to satisfy the need of an organisation at different points of time, without changing the original network.

Applications of computer network

a) **Electronic Mail** (e-mail or Email). The most widely used network application is E-mail, which is forwarding of electronic files to an electronic post office for the recipient to pick up.

b) Scheduling programs allow people across the network to schedule appointments directly by calling up their fellow worker's schedule and selecting a time!

c) Videotext is the capability of having a two-way transmission of picture and sound. Games like distance education lectures, etc. use videotext.

d) Groupware is the latest network application. It allows user groups to share documents, schedules databases

e) Teleconferencing allows people in different regions to "attend" meetings using telephone lines.

f) Automated Banking Machines allow banking transactions to be performed everywhere: at grocery stores, drive-in machines etc.

g) Information Service Providers provide connections to the Internet and other information services.

h) Telecommuting allows employees to perform office work at home by "Remote Access" to the network.

i) Value Added Networks are common carriers such as ERNET, Satyam ,VSNL etc. (they can be private or public companies) who provide additional leased line

connections to their customers. These can be Frame Relay, ATM (Asynchronous Transfer Mode), X.25, etc.

(j) Marketing and sales Marketing professionals use computer network to collect, exchange and analyse data relating to customer needs.

3.3 THE TOPOLOGIES

The topology is the geometric arrangement (either physically or logically) of the linking devices (usually called nodes) and the links, connecting the individual computers or nodes together. Five basic topologies:

1) Bus topology

2) Ring topology

3) Star topology

4) Mesh topology

5) Combined topologies.

1) **The Bus Topology**

In the bus topology there is a single bus that carries all the data to the entire network. A bus is a single continuous communication cable to which all the computers are connected. A cable or bus runs throughout the office to which all the workstations are connected. The bus topology is also known as *linear bus.*

When one workstation wants to talk to another the message or signal travels down the bus in both directions. Each one reads the message to see if it matches its address. The bus topology is a passive topology. It means that the computers connected to the bus amplify the signal on the bus.

The main advantage of bus topology is that it is quite easy to set up. Any workstation can be easily moved to another location as bus runs throughout the office. Another benefit of this layout is that if one computer on the bus fails, it does not affect the rest of the traffic on the bus.

A network with bus topology cannot become too big as all the traffic is on a single bus. The entire network can be down only if the bus has a break. The open ends of bus must be terminated to prevent signal bounce. If one or both ends of the bus are not terminated, the whole network can be down.

Disadvantages include difficult reconfiguration and fault isolation

2) **The Ring Topology**

In the ring topology all the workstations are connected in the shape of a ring. The ring does not have an end. It is made up of short segments that connect one PC to the next and so on, until all the computers are joined in a circle. The signals travel only in one direction and from one PC to the next until it reaches the appropriate node. It is also difficult to move a workstation or to add more computers to an existing ring.

In ring topology the wiring for a ring could be arranged in a circle throughout a building or a group of buildings. The signal travels in one direction only from one computer to the next. The ring topology is an active topology. Each computer boosts the signal (like a repeater) and passes to the next computer till it reaches the destination computer. A drawback of this topology is that if one computer fails, the entire network is down. However, now some ring networks are so designed that a faulty workstation is automatically bypassed. Another drawback is that the traffic is in only one direction. This topology is not used for a large number of nodes.

3) The Star Topology

In the star topology all the stations are connected to a central computer or hub creating a star configuration. The devices are not directly linked to each other Messages pass from the nodes to the hub, where they are processed or passed along to another node. The hub controls the traffic on the network. If the hub fails, the entire network becomes inoperative, but if a node fails it does not affect the rest of the traffic on the network.

All client/server networks use this topology. But since cable from each node must be connected to a central hub, the length of total wiring required increases very much. A hub can be an active hub or a passive hub. A passive hub simply organizes the wiring and works just like a wiring panel for various connections. It does not need any power connection. An active hub does what a passive hub does, but besides this it regenerate and retransmits the signals the way a repeater does. An active hub needs a power connection.

4) Mesh Topology

In a mesh topology, every node has a dedicated point-to point link to every other node. Simply dedicated means that the links carry traffic only between the two nodes. So mesh topology does not have traffic congestion problems every node has n-1 link,for a fully connected mesh topology having n nodes. So the total number of links will be n (n-1).This also means that every node has (n-1) I/O ports.

Advantages of Mesh topology

1) Use of dedicated links guarantees that each connection can carry its own data load. Thus eliminates the traffic problem.

2) If one link fails, it does not affect the rest of network. This means it is robust.

3) Point to point links make fault identification and fault isolation easy.

4) Privacy or security is high, since the other link cannot gain access to the dedicated link where the message is travelling.

Disadvantages of mesh topology

1) More cabling and I/O ports are required, because every node must be connected to every other node.

2) Cost is very high, because more number of node and cabling required.

3) Installation and reconfiguration is difficult.

5) Combined Topologies

A network does not have to stick with one topology. Any two topologies or all the topologies can be used in a network. For example, a hub may be connected to other hubs using a bus and the workstations may be connected by a star.

Two main hybrid topologies are:

1) The Star Bus Topology

The star bus topology is a combination of bus and star topologies. In this topology the hubs of many star topology networks are linked together with a linear bus or trunk. For example, we want to link three star topology networks together. In each network, the nodes are connected to its own hub. Thus we have three hubs. These three hubs are connected by a bus topology.

2) The Star Ring Topology

In this topology the hubs of many star topology networks are connected to another main hub in a star pattern. Thus if we have three star topology networks, then the three hubs of the networks are connected to a fourth hub (main hub) in star pattern.

3.4 CHARACTERISTICS OF THE OSI LAYERS

There are primarily two architectural models for designing computer networks. OSI model & TCP/IP model. In this section we shall discuss one such model.

The seven layers of the OSI reference model can be divided into two categories: upper layers and lower layers.

The *upper layers* of the OSI model deal with application issues and generally are implemented only in software. The highest layer, the application layer, is closest to the end user. Both users and application layer processes interact with software applications that contain a communications component. The term upper layer is sometimes used to refer to any layer above another layer in the OSI model.

The *lower layers* of the OSI model handle data transport issues. The physical layer and the data link layer are implemented in hardware and software. The lowest layer, the physical layer, is closest to the physical network medium (the network cabling, for example) and is responsible for actually placing information on the medium.

Figure 1 illustrates the division between the upper and lower OSI layers.

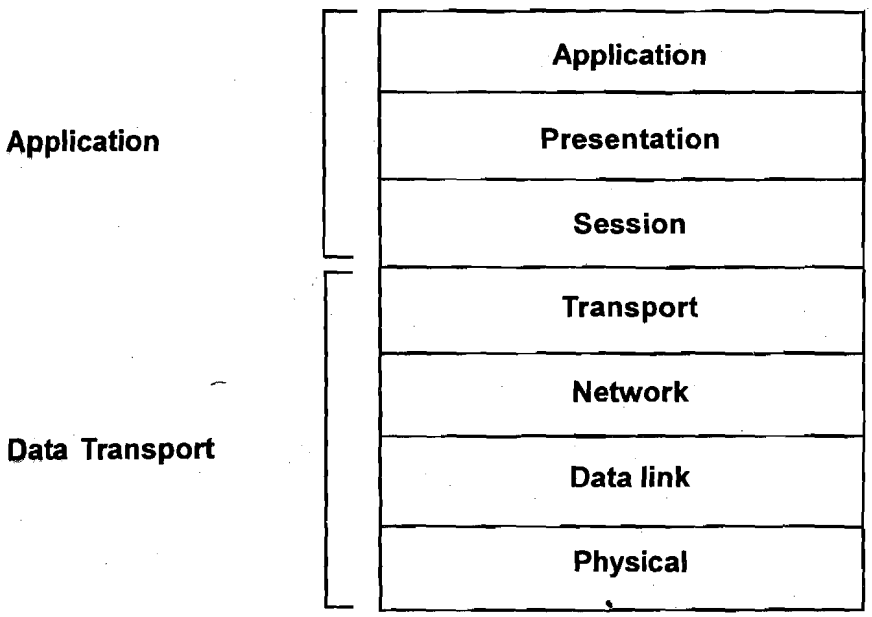


Figure 1: Two Sets of Layers Make Up the OSI Layers

Each layer has well defined functionalities and standard protocols for implementing thes functionalities.

3.5 OSI MODEL AND COMMUNICATION BETWEEN SYSTEMS

Information being transferred from a software application in one computer system to a software application in another must pass through the OSI layers. For example, if a software application in System A has information to transmit to a software application

in System B, the application program in System A will pass its information to the application layer (Layer 7) of System A. The application layer then passes the information to the presentation layer (Layer 6), which relays the data to the session layer (Layer 5), and so on down to the physical layer (Layer 1). At the physical layer, the information is placed on the physical network medium and is sent across the medium to System B. The physical layer of System B removes the information from the physical medium, and then its physical layer passes the information up to the data link layer (Layer 2), which passes it to the network layer (Layer 3), and so on, until it reaches the application layer (Layer 7) of System B. Finally, the application layer of System B passes the information to the recipient application program to complete the communication process.

3.6 INTERACTION BETWEEN OSI MODEL LAYERS

A given layer in the OSI model generally communicates with three other OSI layers: the layer directly above it, the layer directly below it, and its peer layer in other networked computer systems. The data link layer in System A, for example, communicates with the network layer of System A, the physical layer of System A, and the data link layer in System B. *Figure 2* illustrates this example.

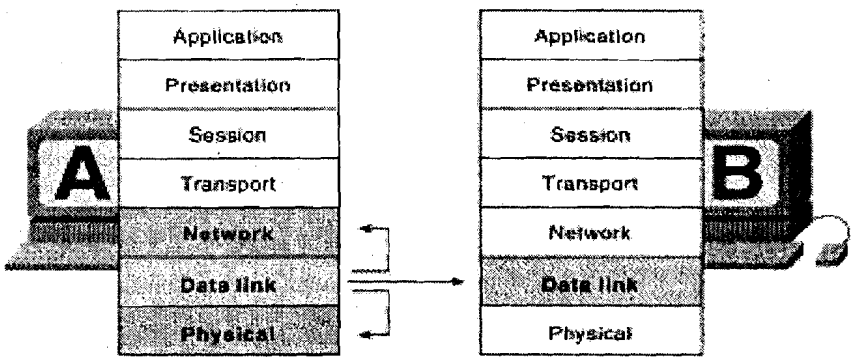


Figure 2: OSI Model Layers Communicate with Other Layers

3.7 PROTOCOLS

Just like human beings need to have a common languages to speak to one another, digital devices and computers also need to have common 'languages' to be able to communicate with one another. The binding function of 'common language' in digital communication is performed by communication protocols.

A communication protocol is a set of conventions or rules that must be adhered to by both communicating parties to ensure that information being exchanged between two parties is received and interpreted correctly. Without a protocol, two devices may be connected but not communicating, just as a person speaking Hindi cannot be understood by a person who speaks only Tamil.

A protocol defines the following three aspects of communication.

1) **Syntax:** The format of data being exchanged, character set used, type of error correction used, type of encoding scheme (e.g., signal level) being used. For example, a simple protocol may use first eight bit for address of sender, the second eight bit for address of receiver and the rest of bit for message itself.

2) **Semantics:** Type and order of messages used to ensure reliable and error free information transfer.

3) **Timing:** Define data rate selection and correct timing for various events during data transfer. Simply when data should be sent and how fast they can be sent.

It has been accepted that the complexity of writing communication software can be reduced by adapting the principle of protocol layering. The idea here is to partition communication functions into a vertical set of layers. Each layer performs a related set of functions. Division of work between layers is done in such a way that they are manageable and provide a logical interface and break point. Each communication layer provides certain services to layers above it and relies on the next lower layer to perform more primitive functions. Each layer hides internal details from other layers. Thus dividing the communication problem into several layers reduces its complexity and makes the work of developing communication software a lot easier and error free.

3.8 TYPES OF NETWORKS

The differences among different types of computer networks are usually based on perspective. For example, computer networks are frequently classified by the geographical area (LAN, MAN, WAN), their topologies (e.g., point to point or broadcast),or the type of communication path they use and the manner in which data are transmitted across this path (e.g., circuit-switched and packet-switched).

Computer networks are classified by the geographical area are:

1) Local Area Networks (LANs)

2) Metropolitan Network (MANs)

3) Wide Area Networks (WANs)

3.8.1 Local Area Networks (LANs)

LANs (local area networks), as shown in *Figure 3* are privately-owned networks that connect computers and resources together in a building or buildings that are close together. LAN plays an important part in everyday functioning of schools, businesses, and government.

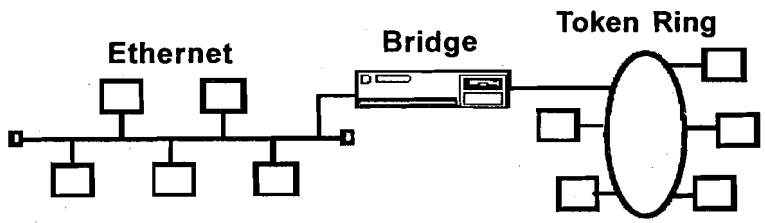


Figure3: Local Area Network in a building

A *LAN* is a high-speed data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, servers, and other devices, so that devices can communicate with each other to share resources. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications.

A Local Area Network is a system of computers that share resources such as disk drives, printers, data, CPU power, fax/modem, applications, etc. They usually have distributed processing, which means that there are many desktop computers distributed around the network and that there is no central processor machine (mainframe).

Location: In a building or individual rooms or floors of buildings or nearby buildings. Can be campus wide like a college or university.

LAN Characterisation

There are four key areas that characterise a local area network.

These are:

1) Transmission Medium
2) Access Method
3) Topology
4) Signaling Techniques

LAN Media-Access Methods

Media contention occurs when two or more network devices have data to send at the same time. Because multiple devices cannot talk on the network simultaneously, some type of method must be used to allow one device access to the network media at a time. This is done in two main ways: carrier senses multiple accesses collision detect (CSMA/CD) and token passing.

In networks using CSMA/CD technology such as Ethernet, network devices contend for the network media. When a device has data to send, it first listens to see if any other device is currently using the network. If not, it starts sending its data. After finishing its transmission, it listens again to see if a collision occurred. A collision occurs when two devices send data simultaneously. When a collision happens, each device waits a random length of time before resending its data. In most cases, a collision will not occur again between the two devices. Because of this type of network contention, the busier a network becomes, the more collisions occur. This is why performance of Ethernet degrades rapidly as the number of devices on a single network increases.

In token-passing networks such as Token Ring and FDDI, a special network packet called a token is passed around the network from device to device. When a device has data to send, it must wait until it has the token and then send its data. When the data transmission is complete, the token is released so that other devices may use the network media. The main advantage of token-passing networks is that they are deterministic. In other words, it is easy to calculate the maximum time that will pass before a device has the opportunity to send data. This explains the popularity of token-passing networks in some real-time environments such as factories, where machinery must be capable of communicating at determinable intervals.

For CSMA/CD networks, switches segment the network into multiple collision domains. This reduces the number of devices per network segment that must contend for the media. By creating smaller collision domains, the performance of a network can be increased significantly without requiring addressing changes.

Normally CSMA/CD networks are half-duplex, meaning that while a device sends information, it cannot receive at the same time. While that device is talking, it is incapable of also listening for other traffic. This is much like a walkie-talkie. When one person wants to talk, he presses the transmit button and begins speaking. While he is talking, no one else on the same frequency can talk. When the sending person is finished, he releases the transmit button and the frequency is available to others.

When switches are introduced, full-duplex operation is possible. Full-duplex works much like a telephone—you can listen as well as talk at the same time. When a network device is attached directly to the port of a network switch, the two devices may be capable of operating in full-duplex mode. In full-duplex mode, performance can be increased, but not quite as much as some like to claim.. However, full-duplex operation does increase the throughput of most applications because the network

media is no longer shared. Two devices on a full-duplex connection can send data as soon as it is ready.

Token-passing networks such as Token Ring can also benefit from network switches. In large networks, the delay between turns to transmit may be significant because the token is passed around the network.

LAN Transmission Methods

For Transmission, LAN usually broadcast their message to all hosts on the LAN The address in the packet or frame enables the destination to receive the packet, while the rest of the hosts ignore the broadcast message.

LAN data transmissions fall into three classifications: unicast, multicast, and broadcast. In each type of transmission, a single packet is sent to one or more nodes.

In a unicast transmission, a single packet is sent from the source to a destination on a network. First, the source node addresses the packet by using the address of the destination node. The package is then sent onto the network, and finally, the network passes the packet to its destination.

A multicast transmission consists of a single data packet that is copied and sent to a specific subset of nodes on the network. First, the source node addresses the packet by using a multicast address. The packet is then sent into the network, which makes copies of the packet and sends a copy to each node that is part of the multicast address.

A broadcast transmission consists of a single data packet that is copied and sent to all nodes on the network. In these types of transmissions, the source node addresses the packet by using the broadcast address. The packet is then sent on to the network, which makes copies of the packet and sends a copy to every node on the network

Topology: The most common LAN topologies are bus, ring, and star.

3.8.2 Metropolitan Networks (MANs)

Metropolitan Area Networks (MANs), as shown in *Figure 4*, are networks that connect LANs together within a city.

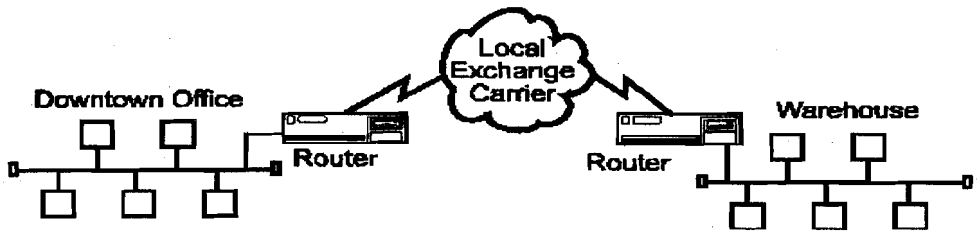


Figure 4: MANs use Local Exchange Carriers

The main criterion for a MAN is that the connection between LANs is through a local exchange carrier (the local phone company) . The protocols that are used for MANs are quite different from those used for LANs (except for ATM, which can be used for both under certain conditions). It has been distinguished as a separate type of network; because of the specific standard known as Distributed Queue Double Bus (DQDB) that has been adopted for MAN. The DQDB comprises two unidirectional buses for connecting computers.

A Metropolitan Area Network is a system of LANs connected throughout a city (*Figure 5*) or metropolitan area. MAN can be considered as a bigger version of a

LAN, typically covering a city. It can be either public or privately owned. MANs have the requirement of using telecommunication media such as voice channels or data channels. Branch offices are connected to head offices through MANs. Examples of organizations that use MANs are universities and colleges, hotels, and banks.

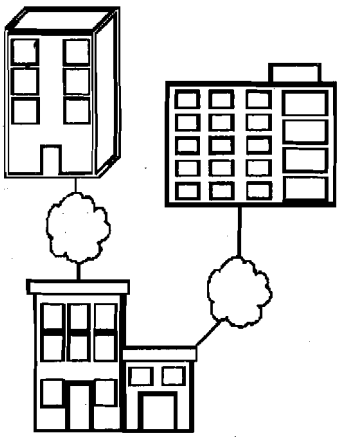


Figure 5: Location: Separate buildings distributed throughout a city

3.8.3 Wide Area Networks (WANs)

Wide Area Networks (WANs) connect LANs together between cities (*Figure 6*).

Communication is usually done through public communication systems such as telephone line, fiber optic cable or wireless technology.

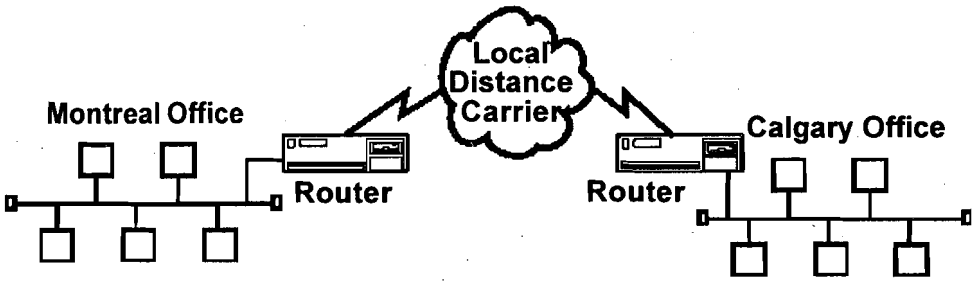


Figure 6: WANs use Long Distance Carriers

A Wide Area Network is a network system connecting cities, countries, or continents together *Figure 7*. WANs are connected together using one of the telecommunications media.

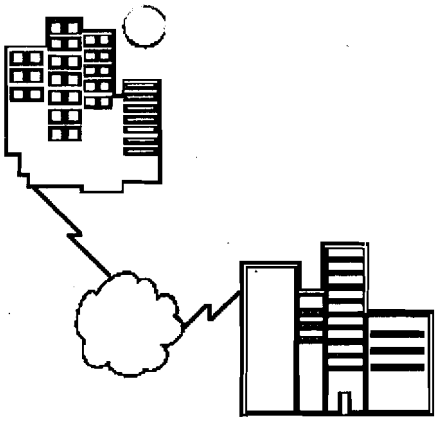


Figure 7: City to city, across a country or across a continent.

The main difference between a MAN and a WAN is that the WAN uses Long Distance Carriers. Otherwise the same protocols and equipment are used as in MAN.

Main differences between a LAN and a WAN are given in the following Table1.

Table1: Difference between LAN and MAN

Wide area Network	Local Area Network
Distance up to thousands of Kilometer	Within a local site
Typical data rates between 9.6k to 1Mbps	High band width between 1-16 Mbps
Higher error rates (1 in 10^5)	Lower error rate (1 in 10^9)
Often use analog circuits from the telephone systems	Use digital signaling over private cables
	Generally use bus or ring topology
Generally has point–to-point link with common topologies mesh and star	Managed by the same company which owns the computers connected to LAN
May be managed by organizations independent of users	LAN uses simple protocols and does not employ any retransmission strategy for lost frames
WAN uses complex protocols and extensive error recovery mechanisms	Number of host on a LAN is limited (usually up to 1024)
Number of node computers has no theoretical limit and could be very large. The practical limit comes from addressing schemes used to identify individual system on the network and other resource constrains.	

3.9 MEDIUM

Data communication system is made up of following components (*Figure 8*).

Figure 8: Data Communication-I

Source Sender: Sender can be Terminals, Computers, Mainframes, workstation, telephone hand set, video camera. Main function of sender is to send data (message) to receiver.

The communications stream through which the data is being transmitted. Examples are:

- Cables
- Microwave Link
- Fiber optic Link
- Radio Frequencies (RF)
- Infrared Wireless

Receiver: The receiver receives the message (data) from sender. Receiver can be Terminals, Computers, Mainframes, workstation, telephone hand set, printer, television and so on.

Protocol: A communication protocol is a set of conventions or rules that must be adhered to by both communicating parties to ensure that information being exchanged between two parties is received and interpreted correctly.

Message: Message consists of text, numbers, pictures, sound, or video – or any combination of these to be transmitted from sender to receiver.

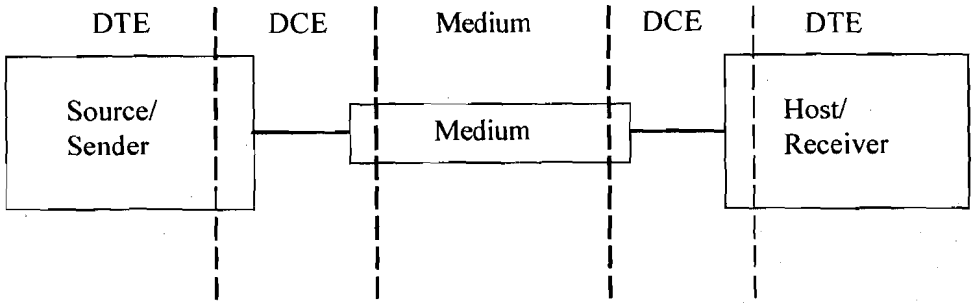


Figure 9: Data Communication-II

DCE: The interface between the Source and the Medium, and the Medium and the Receiver is called the DCE (Data Communication Equipment) (*Figure 9*) and is a physical piece of equipment.

DTE: Data Terminal Equipment is the telecommunications name given to the source and receiver's equipment. It is any device that is a source of or destination for binary digital data.

The DTE generates the data and passes them, through DCE. The DCE takes the generated data by DTE and converts them to an appropriate signal. Then this signal is introduced to telecommunication link. Most commonly used DCE is a modem, discussed in the section.

3.10 DATA FLOW

Data flow (transmission mode) is the flow of data between two points.

There are three types of dataflow *(Figure 10)*: simplex, half duplex and full duplex.

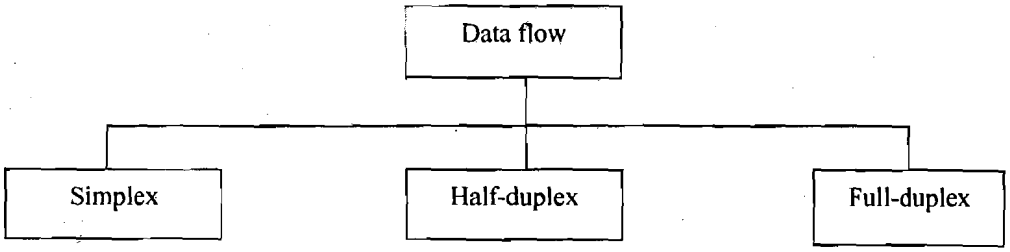


Figure 10: Data Flow

Simplex: data flows in only one direction (Figure 11) on the data communication line (medium). Examples are radio and television broadcasts. They go from the TV station to your home television.

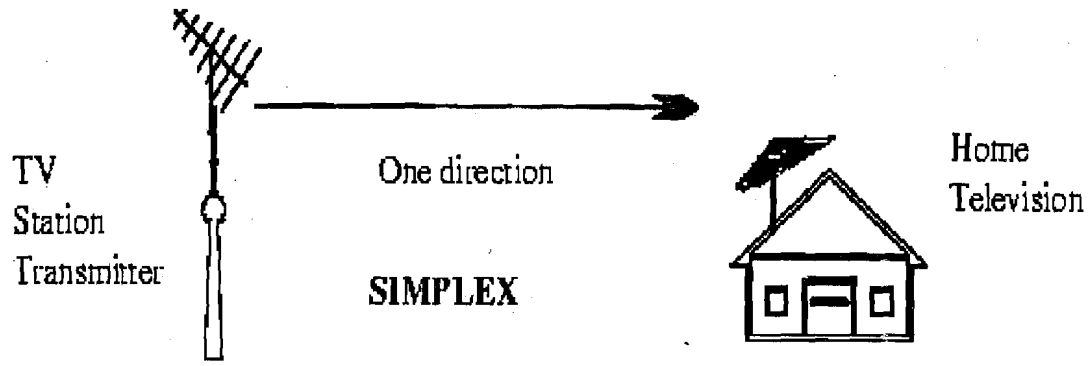


Figure 11: Simplex

Half-Duplex: Data flows in both directions but only one direction at a time (*Figure 12*) on the data communication line. Each of the stations can both transmit and receive. For example; a conversation on walkie-talkie is a half-duplex data flow. Each person takes turns talking. If both talk at once - nothing occurs!

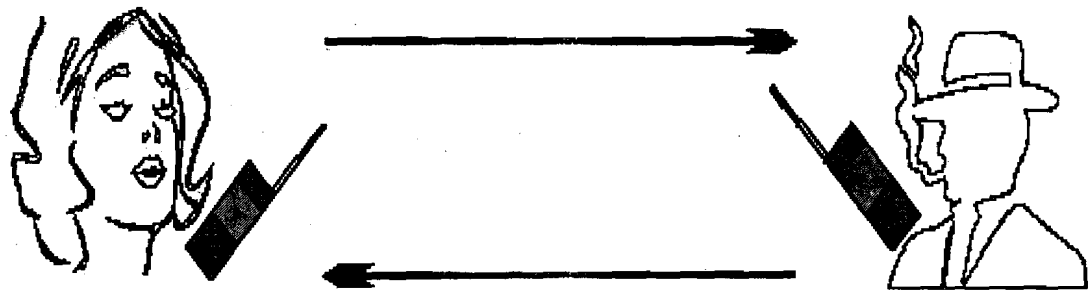


Figure 12: Half-Duplex

Bi-directional but only one direction at a time!

HALF-DUPLEX

Full-Duplex: data flows in both directions at the same time (*Figure 13*). The system is configured to flow data in both directions.

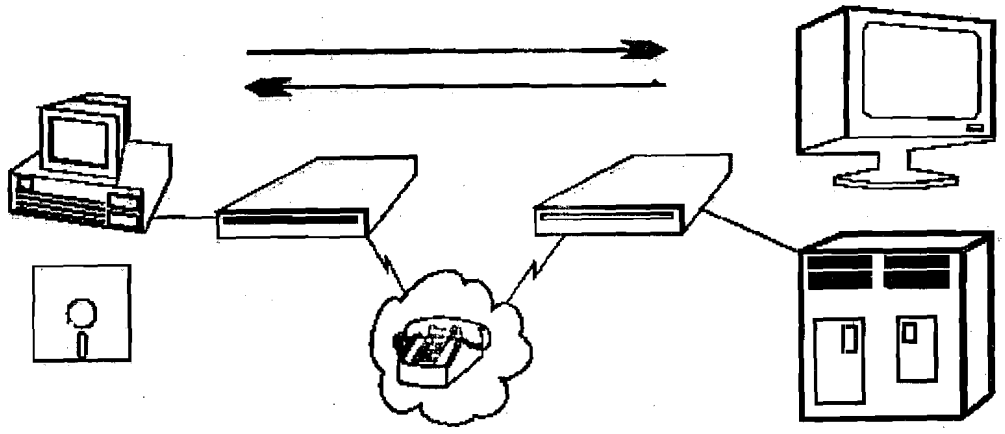


Figure 13: Full Duplex

Bi-directional both directions simultaneously!

Another example of full duplex is two-way street with traffic flowing in both directions, at the same time.

Modems

A modem (MOdulator/DEModulator) connects a terminal/computer (DTE) to the Voice Channel (dial-up line).

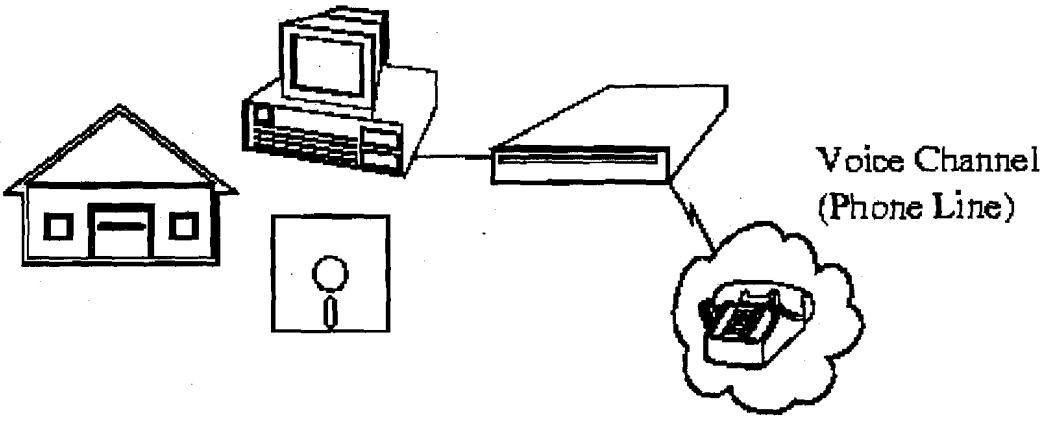


Figure 14: Modems

Basic Definition

The modem (DCE - Data Communication Equipment) is connected between the terminal/computer (DTE - Data Terminal Equipment) and the phone line (voice channel). A modem converts the DTE (Data Terminal Equipment) digital signal to an analog signal (or vice versa) that the voice channel can use.

A modem is connected to the terminal/computer's RS-232 serial port (25 pin male D connector) and the outgoing phone line with an RJ11 cable connector (the same as on a telephone extension cord). Male connectors have pins, female connectors have sockets.

Digital Connection

The connection between the modem and terminal/computer is a digital connection. A basic connection consists of a Transmit Data (TXD) line, a Receive Data (RXD) line and many hardware handshaking control lines (*Figure 15*).

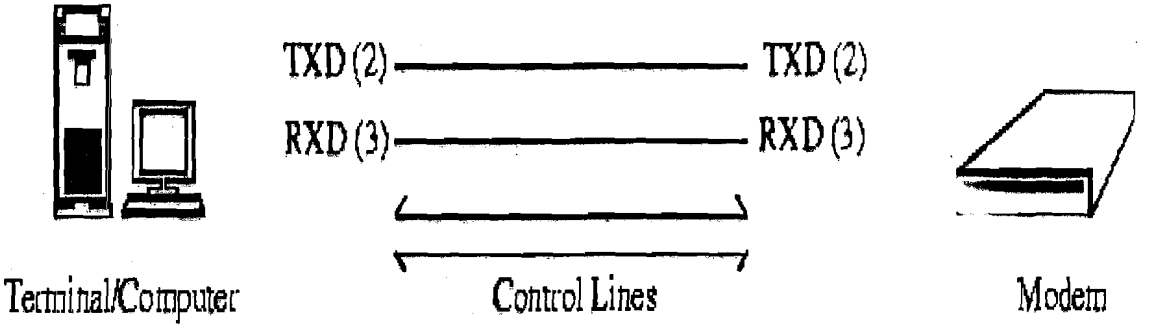


Figure 15: Digital Connection

The control lines determine whose turn it is to talk (modem or terminal), if the terminal/ computer is turned on, if the modem is turned on, if there is a connection to another modem, etc.

Analog Connection

The connection between the modem and the outside world (the phone line) is an analog connection (*Figure 16*). The voice channel has a bandwidth of 0-4 kHz but only 300 - 3400 Hz is usable for data communications.

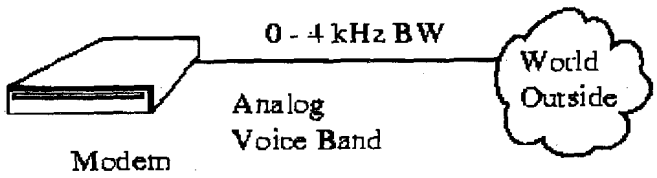


Figure 16: Analog Connection

The modem converts digital information into tones (frequencies) for transmitting through the phone lines.

External/Internal Modems

There are 2 basic physical types of modems: Internal & External modems. External modems *(Figure 17)* sit next to the computer and connect to the serial port using a straight-through serial cable.

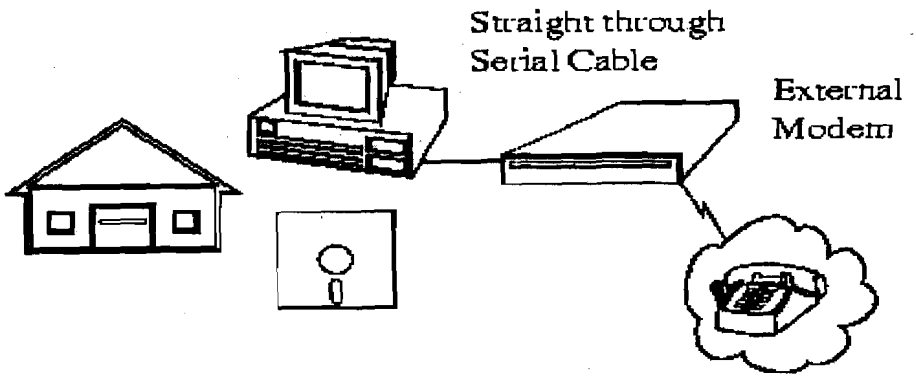


Figure 17: External Modems

Internal modems (*Figure 18*) are a plug-in circuit board that sits inside the computer. It incorporates the serial port on-board. They are less expensive than external modems because they do not require a case, power supply and serial cable. They appear to the communication programs as if they were an external modem for all practical purposes.

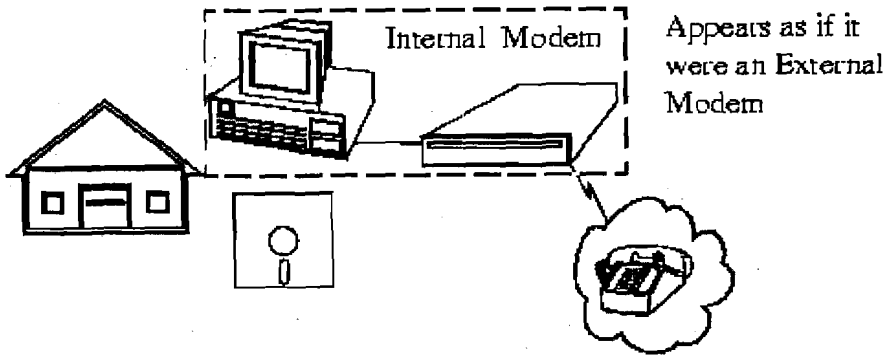


Figure 18: Internal Modems

Modem Types

There are many types of modems, the most common of which are:

(1) Optical Modem: Uses optical fiber cable instead of wire. The modem converts the digital signal to pulses of light to be transmitted over optical lines (more commonly called a media adapter or transceiver).

(2) Short Haul Modem: A modem used to transmit data over 30 km or less. Modems we use at home or to connect computers together among different offices in the same building are short haul modems.

(3) Acoustic Modem: A modem that couples to the telephone handset with what looks like suction cups that contain a speaker and microphones. Used by travelling sales people to connect to hotel phones.

(4) Smart Modem: A modem with a CPU (microprocessor) on board that uses the Hayes AT command set. This allows auto-answer & dial capability rather than manually dialing & answering.

(5) Digital Modem: Converts the RS-232 digital signals to digital signals more suitable for transmission. (Also called a media adapter or transceiver).

(6) V.32 Modem: A milestone modem that uses a 2400-baud modem with 4 bit encoding. This results in a 9600 bps (bits per second) transfer rate.

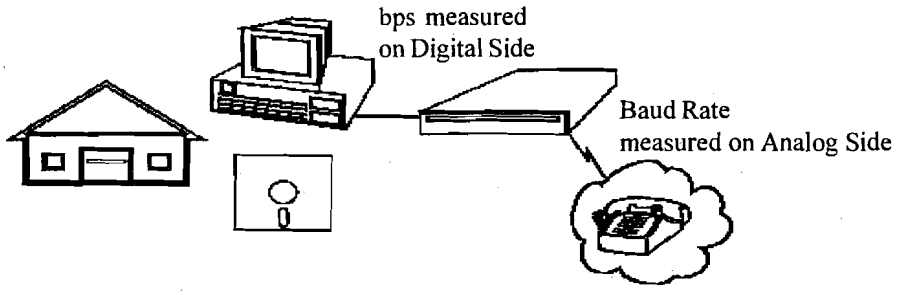


Figure 19: BPS and Baud Rate

Baud (Figure 19) is the speed at which the analog data is changing on the voice channel and bps is the speed at which the decoded digital data is being transferred.

Features of Modems

(1) Speed: The speed at which the modem can send data in bps (bits per second). Typical modem speeds are: 300, 600, 1200, 2400, 4800, 9600, 14.4K, 19.2K, 28.8K bps.

(2) Auto Dial/Re Dial: Smart modems can dial the phone number and auto re dial if a busy signal is received.

(3) Auto Answer: Most modems have Ring Detect capability and can automatically answer the telephone when an incoming call comes in.

(4) Self-Testing: Newer modems have self-testing features. They can test the digital connection to the terminal/computer and the analog connection to a remote modem. They can also check the modem's internal electronics.

(5) Voice Over Data: Voice Over Data modems allow a voice conversation to take place while data is being transmitted. This requires both the source and destination modems to have this feature.

(6) Synchronous or Asynchronous Transmission: Newer modems allow a choice of synchronous or asynchronous transmission of data. Normally, modem transmission is asynchronous (we send individual characters with just start and stop bits). Synchronous transmission or packet transmission is used in specific applications.

3.11 PHYSICAL CONNECTION

The physical connection determines how many bits (1's or 0's) can be transmitted in a single instance of time. If only 1 bit of information can be transmitted over the data transmission medium at a time then it is considered a serial communication (*Figure 20*).

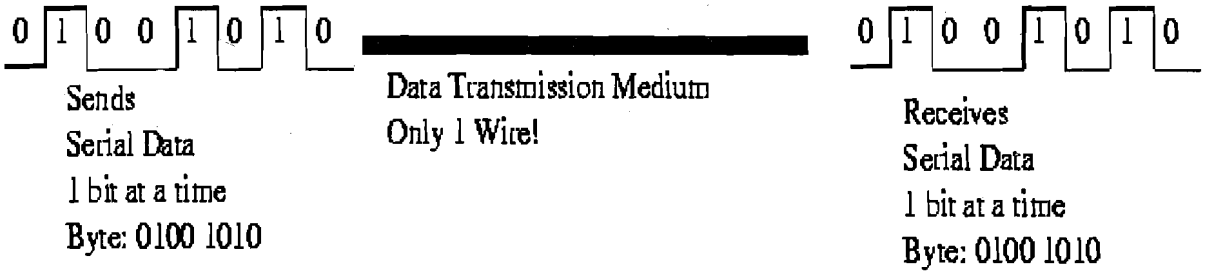


Figure 20: Serial Communication

If more than 1 bit of information is transmitted over the data transmission medium at a time then it is considered a parallel communication (*Figure 21*). By grouping, we can send data n bits at a time instead of one, through n wires.

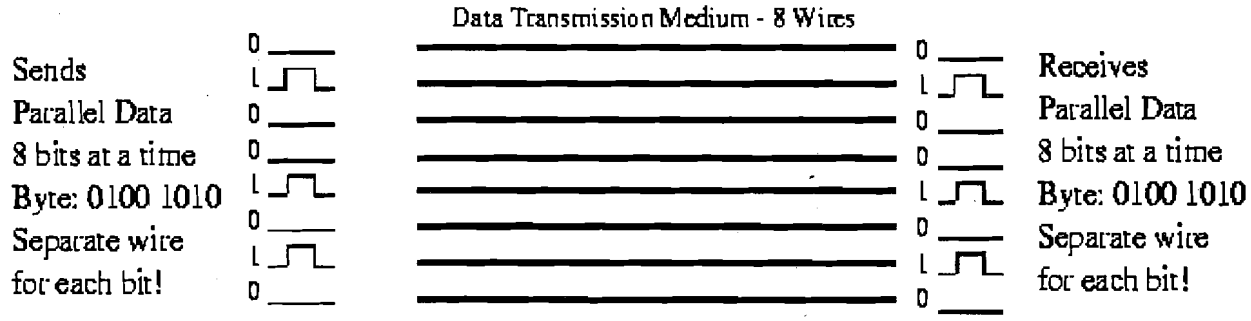


Figure 21: Parallel Communication

Communications	Advantages	Disadvantages
Parallel	Fast Transfer Rates	Short distances only More cost due to more number of lines
Serial	Long Distances	Slow transfer rates Less cost due to only one line required for serialtransmission.

3.12 TRANSMISSION MEDIA

The transmission media provide the physical path for communication among the nodes. In a computer network, where all the nodes are geometrically interconnected, it is known as its topology.

Transmission media can be broadly categorised in to two types: guided and unguided (*Figure 22).*

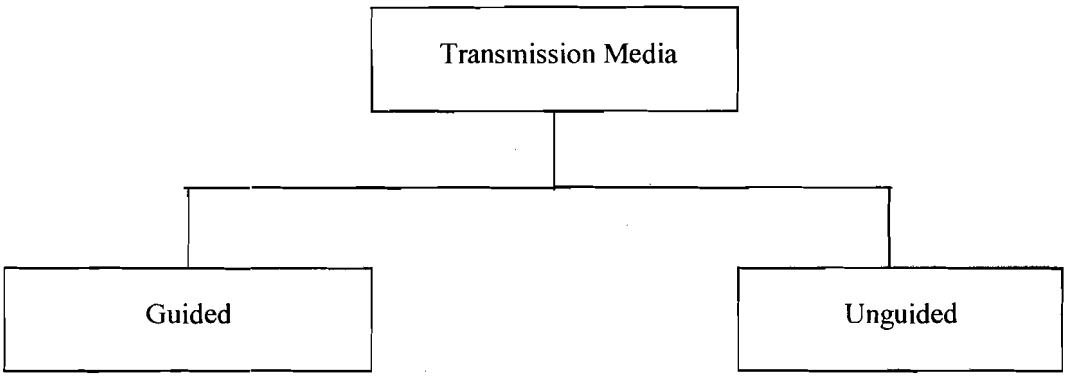


Figure 22: Type of Transmission Media

Guided transmission uses a cabling system that guides the data signals along a specific path. The data signals are bound by the cabling system. Guided media is also known as bound media. "Cabling" is meant in a generic sense, and is not meant to be interpreted as copper wire cabling only. Guided media are commonly used for point-to-point connection. The characteristics of the medium mainly decide the nature and quality of transmission. Guided media are commonly used for LAN application.

Unguided transmission media also called wireless communication consists of a means for the data signals to travel but nothing to guide them along a specific path. The data signals, not bound to a cabling media transport electromagnetic waves without using a physical conductor and are therefore often called unbound media. Unguided media are commonly used for broadcast type communication. Some examples of unguided media are sea water, free space and air. Unguided media are commonly used for WAN application.

Transmission Media Guided

There are 4 basic types of guided media:

a) Open Wire

b) Twisted Pair

c) Coaxial Cable

d) Optical Fiber

Open Wire

Open wire is (*Figure 23*) traditionally used to describe the electrical wire strung along power poles. There is a single wire strung between poles. No shielding or protection from noise interference is used. We are going to extend the traditional definition of open wire to include any data signal path without shielding or protection from noise

interference. This can include multi conductor cables or single wires. This medium is susceptible to a large degree of noise and interference and consequently is not acceptable for data transmission except for short distances of less than 20 ft.

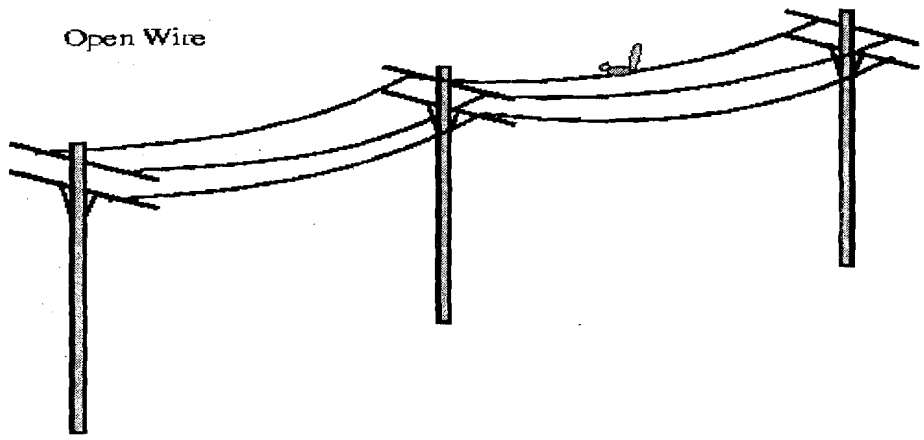


Figure 23: Open Wire

Twisted Pair

The wires in twisted pair (*Figure 24*) cabling are twisted together in pairs. Each pair consists of a wire used for the +ve data signal and a wire used for the -ve data signal. Each pair is twisted together to minimize electromagnetic interference between the pairs.Any noise that appears on 1 wire of the pair will also occur on the other wire. Because the wires are opposite polarities, they are 180 degrees out of phase (180 degrees - phasor definition of opposite polarity). When the noise appears on both wires, it cancels or nulls itself out at the receiving end.

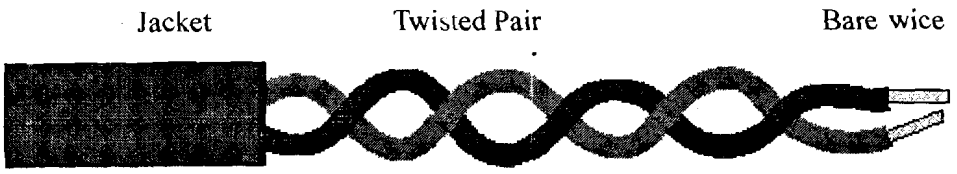


Figure 24: Unshielded Twisted Pair

The degree of reduction in noise interference is determined specifically by the number of turns per foot. Increasing the number of turns per foot reduces the noise interference. To further improve noise rejection, a foil or wire braid "shield" is woven around the twisted pairs. This shield *(Figure 25)* can be woven around individual pairs or around a multi-pair conductor (several pairs).

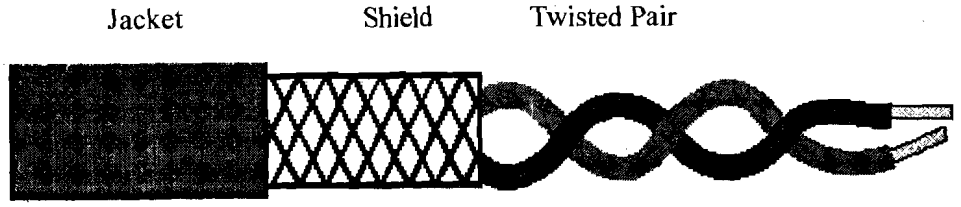


Figure 25: Shielded Twisted Pair

Shielded Twisted Pair

The twisted pair can be shielded (*Figure 25*) with metallic braid, to reduce the interference Cables with a shield are called shielded twisted pair and are commonly abbreviated STP. Cables without a shield are called unshielded twisted pair or UTP. Twisting the wires together results in characteristic impedance for the cable. Typical impedance for UTP is 100 ohm for Ethernet 10BaseT cable.

UTP or unshielded twisted pair cable is used on Ethernet 10BaseT and can also be used with Token Ring. It uses the RJ line of connectors (RJ45, RJ11, etc.)

Use Twisted pair can be used for both analog and digital communication. Twisted pair cables are most effectively used in systems that use a balanced line method of transmission: polar line coding (Manchester Encoding) as opposed to unipolar line coding (TTL logic). Most popular use of twisted pair is in our oldest telephone system. It is also used in LAN for point-to point short distance communication

Coaxial Cable

Coaxial cable (*Figure 26*) consists of two conductors. The inner conductor is held inside an insulator with the other conductor woven around it providing a shield. An insulating protective coating called a jacket covers the outer conductor.

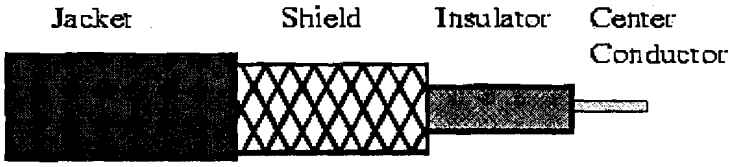


Figure 26: Coaxial Cable

The outer shield protects the inner conductor from outside electrical signals. The distance between the outer conductor (shield) and inner conductor plus the type of material used for insulating the inner conductor determine the cable properties or impedance. Typical impedances for coaxial cables are 75 ohms for Cable TV, 50 ohms for Ethernet Thinnet and Thicknet. The excellent control of the impedance characteristics of the cable allow higher data rates to be transferred than with twisted pair cable.

Optical fiber

Optical fiber consists of thin glass fibers that can carry information at frequencies in the visible light spectrum and beyond. The typical optical fiber consists of a very narrow strand of glass called the core. Around the core is a concentric layer of glass called the cladding. A typical core diameter is 62.5 microns (1 micron = 10^{-6} meters). Typically Cladding has a diameter of 125 microns. Coating the cladding is a protective coating consisting of plastic, it is called the Jacket.

Optical fibers work on the principle that the core refracts the light and the cladding reflects the light. The core refracts the light and guides the light along its path. The cladding reflects any light back into the core and stops light from escaping through it - it bounds the medium.

Advantages of Optical Fiber

- Noise immunity: RFI and EMI immune (RFI - Radio Frequency Interference, EMI -Electromagnetic Interference) because fiber –optic transmission uses light rather than electricity, noise is not a factor.
- Security: cannot tap into cable.

- Large Capacity due to BW (bandwidth).
- No corrosion.
- Longer distances than copper wire.
- Smaller and lighter than copper wire.
- Faster transmission rate.

Disadvantages of optical fiber

- Physical vibration will show up as signal noise!
- Limited physical arc of cable. Bend it too much and it will break!
- Difficult to splice.

The cost of optical fiber is a trade-off between capacity and cost. At higher transmission capacity, it is cheaper than copper. At lower transmission capacity, it is more expensive.

Infrared

Infrared (IR) transmission is another line of sight medium .Infrared technology uses electromagnetic radiation of wave lengths between radio waves and visible light, operation between 100GHZ and 100THZ (terahertz). These frequencies are very high offering nice data transfer rates. We are used to seeing infrared technology utilised for our television or VCR remotes. IR is generally restricted to LAN within or between buildings

Advantages

1) Higher bandwidth means superior throughput to radio
2) Inexpensive to produce
3) No longer limited to tight interroom line-of-sight restrictions

Disadvantage

1) Limited in distance
2) Cannot penetrate physical barriers like walls, ceilings, floors, etc.

3.13 CONNECTING DEVICES

As companies grow, so do their networks. When a network outgrows its original design, the network becomes slow and print jobs take longer to be completed. In such cases it is a better idea to segment the existing LAN so that each segment becomes a separated LAN.

We can connect two or more networks together to create larger networks. A LAN (local area network) can be connected to another LAN. A LAN (local area network) can be connected to another WAN (wide area network). The components or devices that are employed to connect two or more networks together are:

1) Repeaters
2) Hubs
3) Bridges
4) Routers
5) Gateways

3.13.1 Repeaters

Repeaters, also called regenerator, are physical hardware devices. They connect two network segments and broadcast packets between them, thus extending your network beyond the maximum length of your cable segment. They have the primary function to regenerate the electrical signal (shown below):

- Reshaping the waveform
- Amplifying the waveform
- Retiming the signal, to avoid collision on the network

As signal travels along a cable, its strength or amplitude decreases. This is called attenuation. In other words, the signal attenuates as it travels along a cable. This limits the length of a cable used to connect the computers together.

Since signal is a factor in the maximum length of a segment, repeater can regenerate (or amplify) the weak signals so that they can travel additional cable lengths. A repeater has intelligence ,so that it takes a weak signal from one cable segment, regenerates it and passes it on to the next segment. Simply we can say that it recreates the bit pattern of the original signal . No more than four repeaters are used to join segments together to keep collision detection working properly. We should not confuse repeater with amplifiers. As the amplifier uses analog signal, it cannot differentiate between original signal and noise, therefore it amplifies both original signal and noise. The repeater does not amplify the original signal, it regenerates the original bit pattern.

Purpose of a Repeater

The purpose of a repeater (Figure 27) is to extend the LAN Segment beyond its physical limits (as defined by the Physical Layer's Standards: e.g. Ethernet is 500m for 10Base5). A LAN Segment is a logical path, such as the logical bus used by all 802.3 Ethernet types. A LAN Segment is given an identification number, called a Segment Number or Network Number, to differentiate it from other segments.

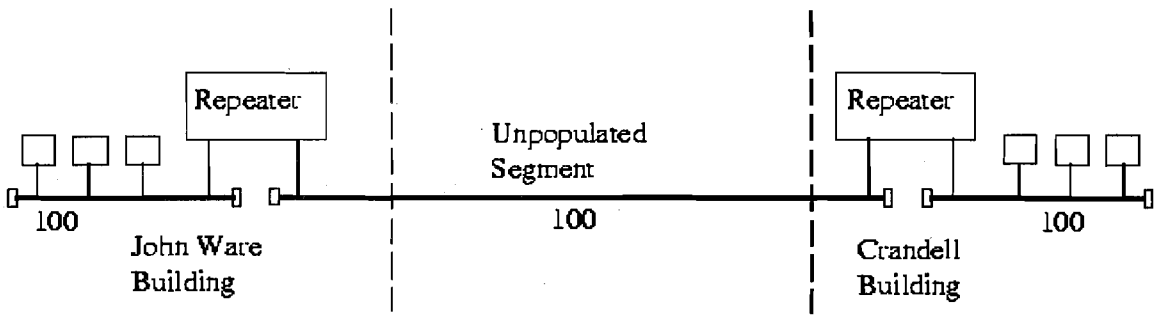


Figure 27: Repeater

Typically, repeaters are used to connect two physically close buildings together (when they are too far apart to just extend the segment). They can be used to connect floors of a building that would normally surpass the maximum allowable segment length. Note: for large extensions, as in the above example, two Repeaters are required. For shorter extensions, only one Repeater may be required.

Repeater's OSI Operating Layer

Repeaters operate at the OSI Model Physical Layer (*Figure 28*).

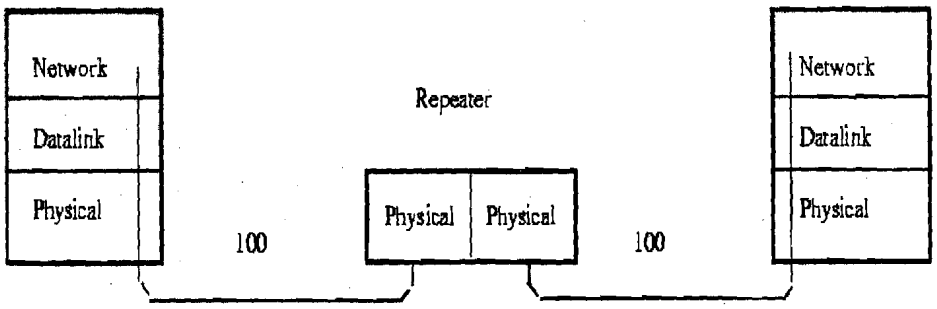


Figure 28: Repeater's Segment-to-Segment Characteristics

A repeater cannot join two cable segments using different access methods. A repeater is not used to connect a segment using CSMA/CD access method to a segment using token passing access. Repeaters can join two different physical media, but they must use the same access method. Thus a repeater can have physical connections to join a coaxial cables segment to a fiber optic segment.

Repeaters do not "de-segment" a network. All traffic that appears on one side of the repeater appears on both sides. Repeaters handle only the electrical and physical characteristics of the signal.

Repeaters work only on the same type of Physical Layer: Ethernet-to-Ethernet, or Token Ring-to-Token Ring. They can connect 10Base5 to 10BaseT because they both use the same 802.3 MAC layer.

You can run into problems with the transfer rate (1 Mbps vs. 10 Mbps) when you connect 1Base5 to 10BaseT. A repeater cannot connect Token Ring to Ethernet because the Physical Layer is different for each network topology.

Repeater Addressing: MAC Layer and Network Segment

The MAC Layer Address is used to identify the Network Card to the Network. The Repeater is transparent to both sides of the segment and both sides can "see" all the Mac Addresses (regardless of which side they are on). This means that any network traffic on Floor 1 will also appear on Floor 5, and vice versa (*Figure 29*).

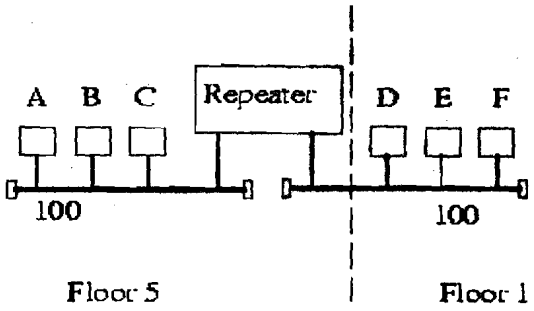


Figure 29: Repeater Addressing

Nodes A & B could be furiously are exchanging files; this network traffic would also appear on Floor 1. Repeaters don't provide isolation between segments (there is only one collision domain).

Because Repeaters provide no isolation between segments, and the repeater is transparent to both sides of the segment, both sides of the repeater appear as one long segment. The Network Number, or Segment Number, is the same on both sides of the Repeater.

3.13.2 Hubs

Hubs can also be called either Multi port Repeaters or Concentrators. They expand one Ethernet connection into many. They are physical hardware devices. A hub is

similar to a repeater, except that it broadcasts data received by any port to all other ports on the hub.

Some hubs are basic hubs with minimum intelligence (i.e. no microprocessors), Intelligent Hubs can perform basic diagnostics, and test the nodes to see if they are operating correctly. If they are not, the Smart Hubs (or Intelligent Hubs) will remove the node from the network. Some Smart Hubs can be polled and managed remotely.

Purpose of Hubs

Hubs are used to provide a Physical Star Topology (*Figure 30*). The Logical Topology is dependent on the Medium Access Control Protocol. At the center of the star is the Hub, with the network nodes located on the tips of the star.

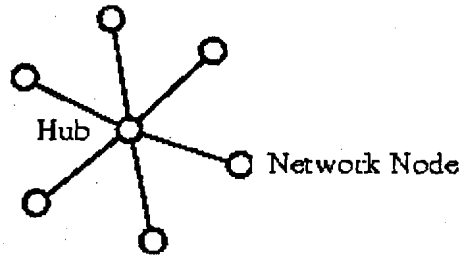


Figure 30: Position of Hub

Star Topology

The Hub is installed in a central wiring closet (Figure 31) with all the cables extending out to the network nodes. The advantage of having a central wiring location is that it is easier to maintain and troubleshoot large networks. All of the network cables come to the central hub. This way, it is especially easy to detect and fix cable problems. You can easily move a workstation in—a star topology— by changing the connection to the hub at the central wiring closet.

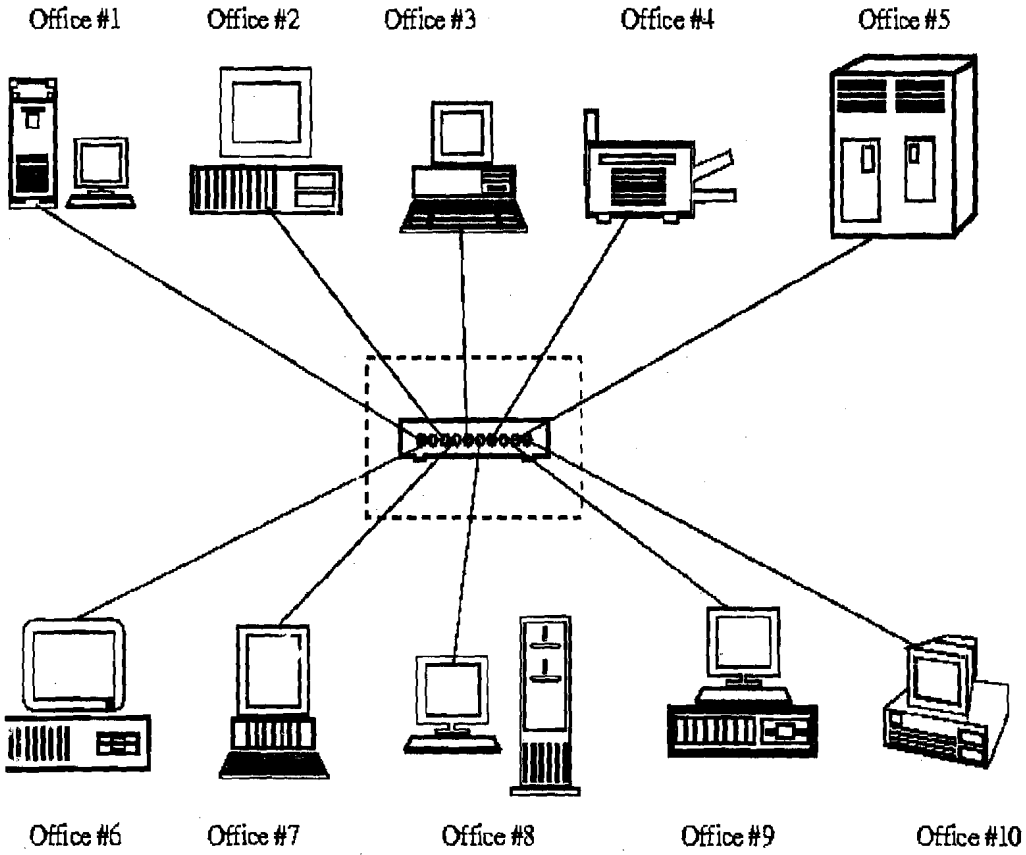


Figure 31: Example of Hub

Hub's OSI Operating Layer

Hubs are multi port repeaters, and as such they obey the same rules as repeaters (See previous section OSI Operating Layer). They operate at the OSI Model Physical Layer.

Hub's Segment-to-Segment Characteristics

To understand the Ethernet segment-to-segment *(Figure 32)* characteristics of a hub, determine how the Ethernet Hubs operate. Logically, they appear as a Bus Topology, and physically as a Star Topology. Looking inside an Ethernet Hub, we can see that it consists of an electronic printed circuit board (which doesn't tell us much). If we form a functional drawing, then we can clearly see how the Physical and Star Topology appears:

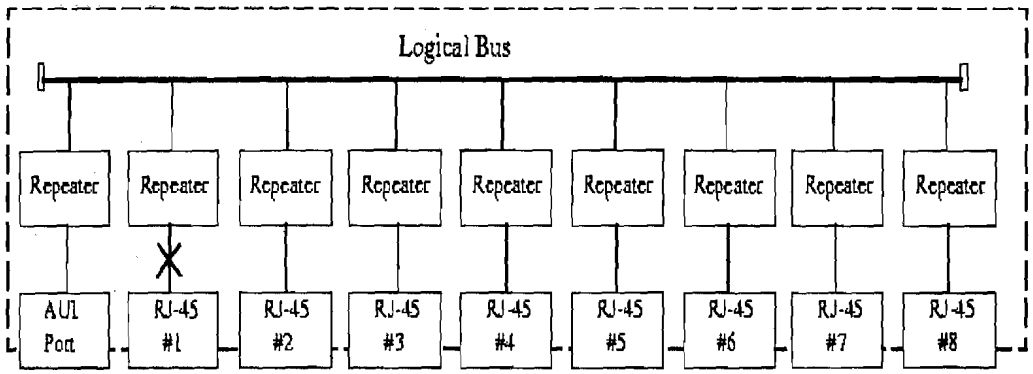


Figure 32: Hub's Segment to Segment

Understanding that inside the Hub is only more repeaters, we can draw the conclusion that all connections attached to a Hub are on the same Segment (and have the same Segment Number). A single repeater is said to exist from any port to any port, even though it is indicated as a path of 2 repeaters.

Hub's Addressing

Again, because a Hub is just many repeaters in the same box, any network traffic between nodes is heard over the complete network. As far as the stations are concerned, they are connected on one long logical bus (wire).

Switching Hubs

Switching hubs (*Figure 33*) are hubs that will directly switch ports to each other. They are similar to full duplex hubs, except that they allow dedicated 10 Mbps channels between ports.

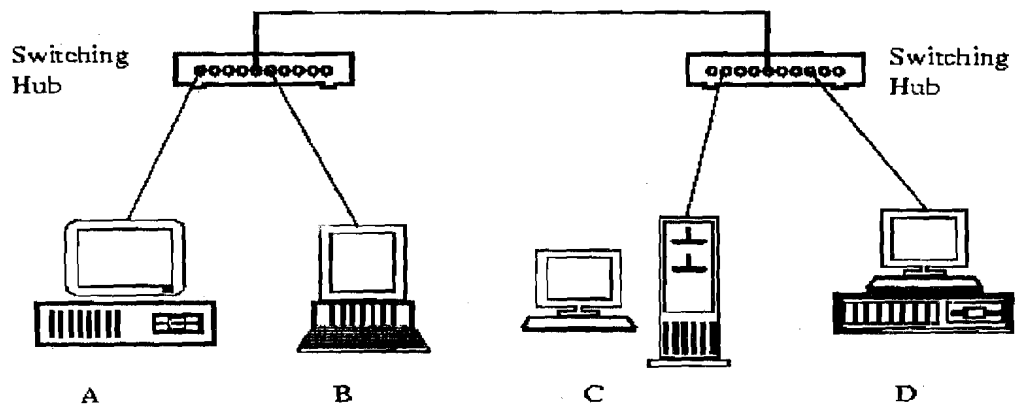


Figure 33: Switching Hub

If A wanted to communicate with B, a dedicated 10 Mbps connection would be established between the two. If C wanted to communicate with D, another dedicated 10 Mbps connection would be established.

3.13.3 Bridges

Bridges have all the features of the repeater. Besides regenerating the signals, a bridge can segment (or divide) a network to isolate traffic related problems. A bridge sends the data frames only to the concerned segment, thus preventing excess traffic. A bridge can split an overloaded network into two separate networks, reducing the amount of traffic on each segment and thus making each network more efficient. Just like repeaters, the bridges can be used to link different physical media. Bridges can also be used to connect dissimilar networks like Ethernet system to a Token Ring system. Thus bridges can be used to join networks using CSMA/CD access and token passing access.

Bridges are both hardware and software devices. They can be standalone devices — separate boxes specifically designed for bridging applications— or they can be dedicated PCs (with 2 NICs and bridging software). Most server software will automatically act as a bridge when a second NIC card is installed.

Bridge OSI Operating Layer

Bridges (*Figure 34*) operate on the OSI Model Data Link Layer, while repeaters work at the physical layer. Since bridges work on a higher layer than repeaters, they are more complex than repeaters and cost more than repeaters. They look at the MAC addresses for Ethernet and Token Ring, and determine whether or not to forward--or ignore—a packet.

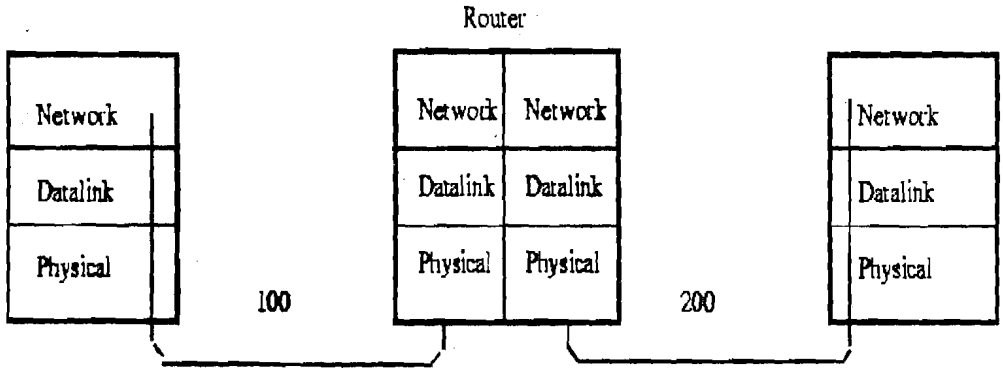


Figure 34: Bridge OSI Operating Layer

Bridges have their own routing tables. Initially the bridge's routing table is empty. As nodes send packets, the source address is copied to the routing table. With this address information, the bridge learns where the computers are situated. When any packet is received by a bridge it reads its source and destination address. If the bridge knows the location of the destination node it forwards the packet to the segment on which the destination node is situated. If it does not know the destination, it forwards the packet to all the segments.

Purposes of a Bridge

The purposes of a Bridge are the following:

- Isolates networks by MAC addresses
- Manages network traffic by filtering packets
- Translates from one protocol to another

Isolates networks by MAC addresses

For example, you have one segment called Segment 100: it has 50 users (in several departments) using this network segment. The Engineering Dept. is CAD (Computer Aided Design) -oriented, while the Accounting Dept. is into heavy number crunching (year end reports, month end statements, etc.).

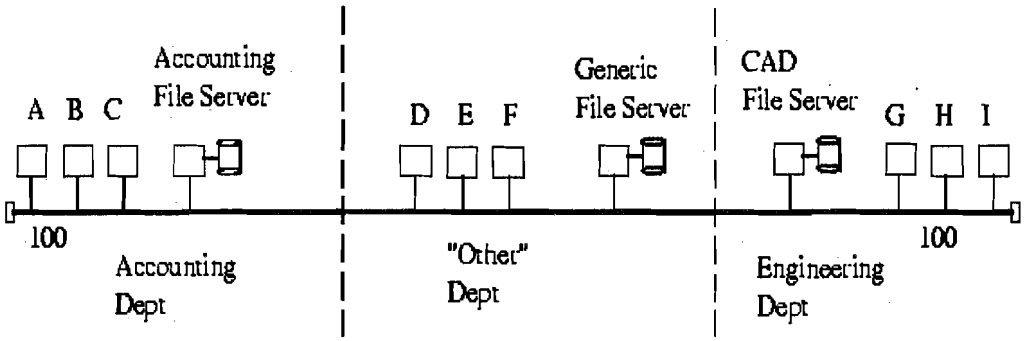


Figure 35: Bridge Example 1

On this network, any traffic between Clients A, B or C and the Accounting File Server (in the Accounting Dept.) will be heard across the Segment 100. Likewise, any traffic between the Engineering Dept. Clients G, H or I (to the CAD File Server) will be heard throughout the Network Segment. The result is that "Other" Department accesses to the Generic File Server are incredibly slow: this is because of the unnecessary traffic that is being generated from other departments (Engineering & Accounting).

The solution is to use one Bridge (*Figure 36) to* isolate the Accounting Dept., and another bridge to isolate the Engineering Department. The Bridges will only allow packets to pass through that are not on the local segment. The bridge will first check its "routing" table to see if the packet is on the local segment. If it is, it will ignore the packet, and not forward it to the remote segment. If Client A sent a packet to the Accounting File Server then Bridge #1 will check its routing table (to see if the Accounting File Server is on the local port). If it is on the local port, then Bridge #1 will not forward the packet to the other segments.

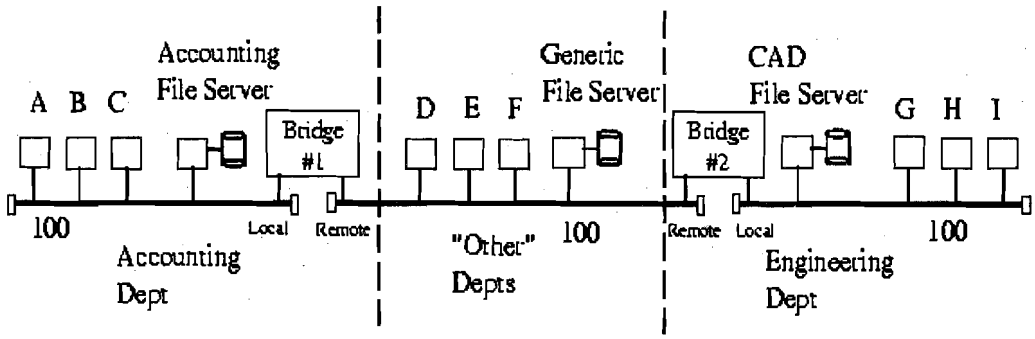


Figure 36: Bridge Example 2

If Client A sent a packet to the Generic File Server, Bridge #1 will again check its routing table to see if the Generic File Server is on the local port. If it is not, then Bridge #1 will forward the packet to the remote port.

Note: The terms local and remote ports are arbitrarily chosen to distinguish between the two network ports available on a bridge.

In this manner, the network is segmented, and the local department traffic is isolated from the rest of the network. Overall network bandwidth increases because the Accounting Dept. does not have to fight with the Engineering Dept. (for access to the segment). Each segment has reduced the amount of traffic on it and the result is faster access. Each department still has complete access to the other segments, but only when required.

3.13.4 Routers

A router is a special –purpose computer having a processor (CPU) and memory like any other computer. But unlike any other computer, it has more than one I/O interface that allows it to connect to multiple computer networks.

Routers are both hardware and software devices. Just like bridges, Router can connect network segments and filter and isolate traffic. Unlike a bridge, a router can connect networks that use different technologies, addressing methods, media types, frame formats, and speeds. Routers are used in complex network situations because they provide better traffic management than bridges. A router keeps track of the address of all the segment of a network and can even determine the best path for sending data. Routers do not pass broadcast traffic.

Like bridges, the routers also maintain routing tables in their memories to store information about physical connections on the network. The router examines each packet of data, checks the routing table , and then forwards the packet if necessary. Routers are more inelegant than bridges, as routers can share status and routing information with one another and use this information to bypass slow or malfunctioning connections. Routers do not maintain any state information about the packets; they simply move them along the network. Routers are usually employed by wide area networks using dissimilar addressing schemes and different communication protocols.

Routers do not allow bad data to get passed on to the network. Thus they save networks from broadcast storms.

There are two types of routers – static routers and dynamic routers.

Static routers require an administrator to manually set up and configure the routing table and to specify each route.

Dynamic routers maintain a routing table automatically and require minimal set up and configuration.

Router OSI Operating Layer

Routers operate on the OSI Model's Network Layer as shown in *Figure 37*. The Internet work must use the same Network Layer protocol. Routers allow the transportation of the Network Layer PDU through the Internetwork, even though the Physical and Data Link Frame size and addressing scheme may change.

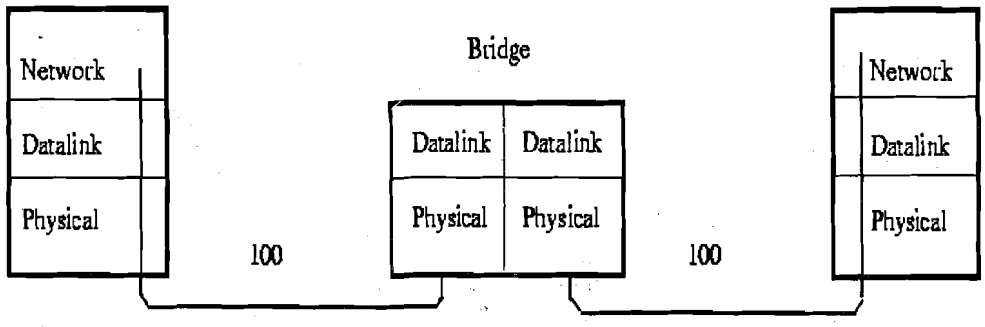


Figure 37: Router Segment-to-Segment Characteristics

Routers that only know Novell IPX (Internetwork Packet Exchange) will not forward Unix's IP (Internetwork Packet) PDUs, and vice versa. Routers only see the Network Layer protocol that they have been configured for. This means that a network can have multiple protocols running on it (e.g. SPX/IPX, TCP/IP, Appletalk, XNS, etc.).

Router Addressing

Routers know the address of all known networks. They maintain a table of pathways between networks and can select an optimal route over which to send data. Routers look only at network address and not at destination node address. Routers talk to other routers, but not to remote computers.

Routers combine the Network Number and the Node Address to make Source and Destination addresses (in routing Network Layer PDUs across a network). Routers have to know the name of the segment that they are on, and the segment name or number where the PDU is going. They also have to know the Node Address: MAC Address for Novell, and the IP address for TCP/IP.

3.13.5 Gateways

A Gateway is the Hardware/Software device that is used to interconnect LANs & WANs

Gateways are much more complex and powerful than a router. They are slower than a router and are expensive. A Gateway incorporates the functions of routers and bridges, but it can translate instruction set on sending network into corresponding instruction set of the receiving network. Gateways make communication possible between different architectures and environments.

Often, the router that is used to connect a LAN to the Internet will be called a gateway. It will have added capability to direct and filter higher layer protocols (layer 4 and up) to specific devices (such as Web servers, ftp servers and e-mail servers).

A Gateway links two systems that do not use the same communication protocols, data formatting structures, languages and architecture, which can not be done by a router. Gateways perform protocol and data conversion.

Gateway's OSI Operating Layer

A Gateway operates at the Transport Layer and above and it typically translates each source layer protocol into the appropriate destination layer protocol. Gateways use all the seven layers of the OSI model A mainframe gateway may translate all OSI Model layers. For example, IBM's SNA (System Network Architecture) does not readily conform to the OSI Model, and requires a gateway to translate between the two architectures.

☞ Check Your Progress 1

1) Compare the advantage of fiber over copper wire.

.....

.....

2) Discuss the advantages and disadvantages of Bus & Mesh Topologies.

.....

.....

3) What are the roles of protocols in a computer network?

.....

.....

3.14 SUMMARY

- A network allows one to share access to information devices.
- Communication protocol is a set of conventions or rules that must be adhered to by both communicating parties to ensure that information being exchanged between the two parties is received and interpreted correctly.
- The major criteria to judge a data communication network are: performance Consistency, Reliability, Recovery, Security.
- The topology is the geometric arrangement (either physically or logically) of the linking devices (usually called nodes) and the links, connecting the individual computers or nodes together. Different topologies are mesh, star, ring or combined topology.
- Communication between two devices can occur in three transmission modes: simplex, half-duplex or full duplex.
- Computer network is classified into three types: LAN, MAN and WAN.
- The network of networks is called the Internet.
- The most familiar type of DCE is the modem that modulates and demodulates signals.
- Guided transmission media use a cabling system that guides the data signals along a specific path.
- Unguided transmission media consist of a means for the data signals to travel but nothing to guide them along a specific path.
- Repeater is a device that operates at the physical layer, bridge at the data link layer, router at the network layer and Gateway at all seven layers of the OSI model.

3.15 SOLUTIONS/ANSWERS

Check Your Progress 1

- Noise immunity: RFI and EMI immune (RFI - Radio Frequency Interference, EMI -Electromagnetic Interference) Because fiber –optic transmission uses light rather than electricity, noise is not a factor.
- Security: cannot tap into cable.
- Large Capacity due to BW (bandwidth).
- No corrosion.
- Longer distances than copper wire.
- Smaller and lighter than copper wire.
- Faster transmission rate.

The cost of optical fiber is a trade-off between capacity and cost. At higher transmission capacity, it is cheaper than copper. At lower transmission capacity, it is more expensive.

2) **The Bus Topology**

The main advantage of bus topology is that it is quite easy to set up. Any workstation can be easily moved to another location as bus runs throughout the office. Another benefit of this layout is that if one computer on the bus fails, it does not affect the rest of the traffic on the bus.

A network with bus topoiogy cannot become too big as all the traffic is on a single bus. The entire network can be down only if the bus has a break. The open ends of the bus must be terminated to prevent signal bounce. If one or both ends of the bus are not terminated, the whole network can be down.

Disadvantages include difficult reconfiguration and fault isolation

Mesh Topologies

In a mesh topology, every node has a dedicated point-to point link to every other node. Simply dedicated means that the links carry traffic only between the two nodes. So mesh topology does not have traffic congestion problems. Every node has n-1 link, for a fully connected mesh topology having n nodes. So total number of links will be n (n-1).This also means that every node has (n-1) I/O ports

Advantages of Mesh topology

1) Use of dedicated links guarantees that each connection can carry its own data load. This eliminates the traffic problem.

2) If one link fails, it does not affect the rest of network. This means it is robust.

3) Point to point links make fault identification and fault isolation easy.

4) Privacy or security is high; as any other link cannot gain access to dedicated link where the message is travailing.

Disadvantages of mesh topology

1) More cabling and I/O ports are required, because every node must be connected to every other node.

2) Cost is very high, because more number of nodes and cabling required.

3) Installation and reconfiguration is difficult.

4) The complexity of writing communication software can be reduced by adopting the principle of protocol layering. The idea here is to partition communication functions into a vertical set of layers. Each layer performs a related set of functions. Division of work between layers is done in such a way that they are manageable and provide a logical interface and break point. Each communication layer provides certain services to layers above it and relies on the next lower layer to perform more primitive functions. Each layer hides internal details from other layers. Thus dividing the communication problem into several layers reduces its complexity and makes the work of developing communication software a lot easier and error free.

3.16 FURTHER READINGS

1) *"Computer Networks, Tanenbaum"*, Third Edition, Prentice-Hall 1996.

2) *"Data and Computer Communications"*, William Stallings, Fourth Edition, MacMillan, 1994.

3) *"Data communication and networkings"*. Behrouz A.Forouzan 2nd edition, TMH 2000.

4) *"Internetworking with TCP/IP", Douglas Comer,* Volume I, Fourth Edition, Prentice Hall, 2000.

UNIT 4 INTERNETWORKING: CONCEPT, ARCHITECTURE AND PROTOCOLS

Structure **Page Nos.**

4.0 INTRODUCTION

An internetwork is a collection of packet-switching and broadcast networks, connected by bridges, switches, or routers which are intermediate networking devices, that functions as a single large network. So all users and devices can communicate, regardless of the network segment to which they are attached. *Figure 1* illustrates some different kinds of network technologies that can be interconnected by routers and other networking devices to create an internetwork.

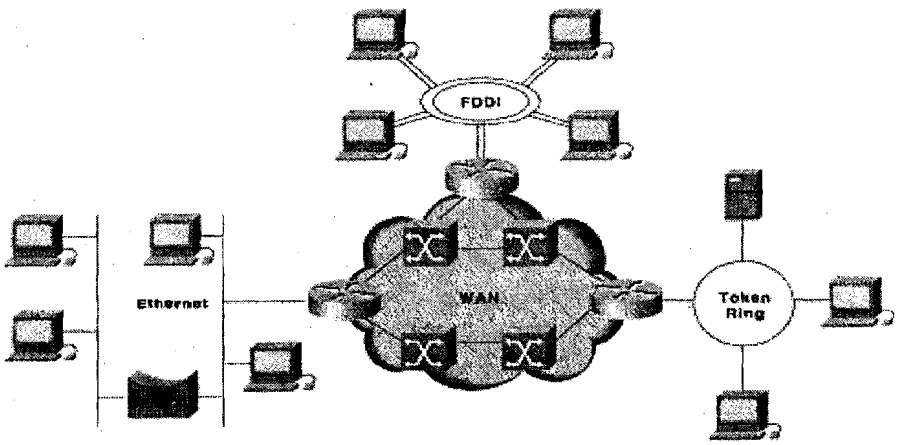


Figure 1: Internetworking

In this unit we will explain the concepts, architecture and protocols of internetworking.

4.1 OBJECTIVES

After going through this unit you should be able to:

- define packet switching concept;
- differentiate between virtual circuit & datagram, and
- understand the functioning of a large number of protocols.

4.2 HISTORY OF INTERNETWORKING

Networking allows computers to share information, applications and even hardware devices. For any two computers to communicate with each other , they must follow a set of rules called protocol. In early years of networking, every vendor was concerned only with his own product. Each vendor defined its own standard for communication between its computers. This resulted in many different types of standards for network hardware and software that did not share a common protocol. This meant that different systems could not interact with each other .Two computers using network products of different vendors could not be connected together.

In the late 1970s, the networking community came together in an effort to replace these closed systems with open systems. They wanted that all networking products should be compatible with other vendor products. The International Standards Organisation (ISO) developed Open Systems Interconnection (OSI) reference model for networking the model gives guidelines about how the parts of a network communication system should work together.

The Open System Interconnection (OSI) reference model describes how information from a software application in one computer moves through a network medium to a software application in another computer. The OSI reference model is a conceptual model composed of seven layers, each specifying particular network functions. The model was developed by the International Standards Organisation (ISO) in 1984, and it is now considered the primary architectural model for inter computer communications. The OSI model divides the tasks involved with moving information between networked computers into seven smaller, more manageable task groups. A task or group of tasks is then assigned to each of the seven OSI layers. Each layer is reasonably self-contained so that the tasks assigned to each layer can be implemented independently. This enables the solutions offered by one layer to be updated without adversely affecting the other layers. The following list details the seven layers of the Open System Interconnection (OSI) reference model:

- Layer 7—Application
- Layer 6—Presentation
- Layer 5—Session
- Layer 4—Transport
- Layer 3—Network
- Layer 2—Data link
- Layer 1—Physical

7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data link
1	Physical

Figure 2 : The OSI Reference Model Contains Seven Independent Layers

4.3 PACKET SWITCHING

Data communication takes place between two devices that are directly connected through some form of transmission medium. It is impractical for two devices to be connected directly. Instead, a network of switching nodes provides a transfer path between two devices. Packet switching involves the breaking up of messages into smaller components called packets. Packets often range in size from about 128 bytes to over 4096 bytes depending on the system involved. Each packet contains source and destination information, and is treated as an individual message. These mini-messages are received and routed through optimal routes by various nodes on a wide area network. For example a file to be transmitted between two machines may be broken into many packets that are sent across the network one at a time. The network H/W delivers the packet to the end where the network software reassembles them into a single file again. There are two major types of packets to be switched. They are **datagram** and **virtual circuit.**

In the **datagram approach,** each packet is treated independently and may follow a different path through the network. Packets may be re-ordered, dropped or delivered in wrong sequence. The communication protocols will have to provide error recovery and sequencing of packets at the destination.

In the **virtual circuit** approach, a fixed logical path through the network from sender to destination is established before any packets are sent. This path remains unchanged for the duration of the connection or session. Although no resources are reserved along the path, packets are buffered at intermediate nodes awaiting transmission.

Thus, a virtual circuit only defines a path for packets to follow without actually reserving dedicated channels along the route as is the case with circuit switching. Virtual circuit may provide a number of services including sequencing, error control and **flow control.**

In comparing datagram and virtual circuit switching with other switching technologies, there are several factors to be considered. First of all, packet switching is faster because messages are not stored in their entirety for later retrieval. Each packet is small enough to be stored in a router's machine memory until it can be routed an instant later. Secondly, packet switching allows the avoidance of route failure due to excessive traffic loads. This is accomplished by routing packets along routes that are the most free and clear. Thirdly, packet switching spreads the load of communication across several paths.

The motivation for adopting packet switching are cost and performance. Because multiple machines can share network fewer interconnections are required and cost is kept low. Packet switched networks that span large geographical distances are fundamentally different from those that span short distances. To help characterize the differences in capacity and intended use, packet switched technologies are often divided into three broad categories: Wide Area Network (WAN), Metropolitan Area Network (MAN) and Local Area Network. There are structural protocols for each category. In the next section we will examine one such protocol.

4.4 INTERNETWORKING CONCEPTS

We have seen how machines connect to individual networks. The question arises, "How are networks interconnected to form an internetwork?" The answer has two parts. Physically, two networks can only be connected by a computer that attaches to both of them. A physical attachment does not provide the interconnection we have in mind, however, because such a connection does not guarantee that the computer will cooperate with other machines that wish to communicate. To have a viable internet, we need computers that are willing to shuffle packets from one network to another. Computers that interconnect two networks and pass packets from one to the other are called internet gateways+ or internet routers.

Consider an example consisting of two physical networks shown in *Figure 3*. In the figure, machine G connects to both network 1 and network 2. For G to act as a gateway, it must capture packets on network 1 that are bound for machines on network 2, and packets on network 2 that are destined for machines on network 1 and transfer them.

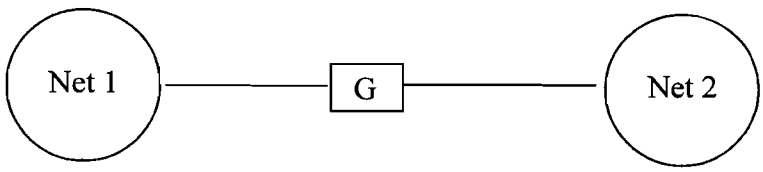


Figure 3: Two networks interconnected by G, a gateway (router)

Interconnection through IP Gateways or Routers

When internet connections become more complex , gateways need to know about the topology of the internet beyond the networks to which they connect. For example, Figure 4 shows three networks interconnected by two gateways.

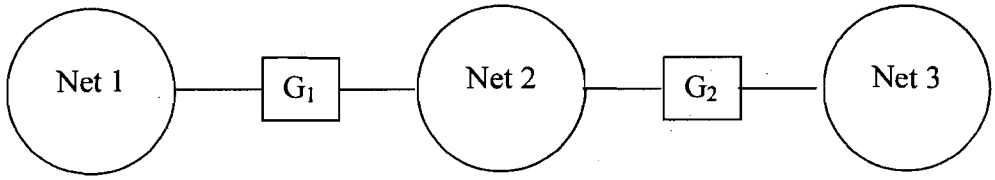


Figure 4: Three networks interconnected by two gateways

In this example, gateway G must move from network 1 to network 2 all packets destined for machines on either network 2 or network 3. As the size of the internet expands, the gateway's task of making decisions about where to send packets becomes more complex.

The idea of a gateway seems simple, but it is important because it provides a way to interconnect networks, not just machines. In fact, we have already discovered the principle of interconnection used throughout an internet:

In a TCP/IP internet, computers called gateways provide all interconnections among physical networks.

You might suspect that gateways, which must know how to route packets to their destination, are large machines with enough primary or secondary memory to hold information about every machine in the internet to which they attach. However, gateways used with TCP/IP internets are usually minicomputers; they often have little or no disk storage and limited main memories. The trick to building a small internet gateway lies in the following concept:

Gateways route packets based on destination network, not on destination host.

If routing is based on networks, the amount of information that a gateway needs to keep is proportional to the number of networks in the internet, not the number of machines.

4.5 INTERNET ADDRESSES

In the previous section we defined a TCP/IP internet as a virtual network built by interconnecting several physical networks through gateways which are usually controlled by the internet service providers think of internet as a large network like any other network. The difference of course is that the internet is a virtual structure imagined by its designers and implemented entirely in TCP/IP s/w. Each machine (host) on an internet is assigned a unique 32 bit internet address that is used in all communication with that machine. Conceptually each address is a paid (netid, hosted) where **netid** identifies a network and hosted identifies a host on that network. Because IP addresses comprise both a network as a host on that network they do not specifiy an individual machine but a connection to a network. For example a getway connecting networks has distinct IP address, one for each network connection.

Unicasting, Broadcasting, and Multicasting

When an IP datagram is sent to an individual IP address, it is called a unicast IP datagram. The process of sending the datagram is called unicasting. Unicasting is used when two IP nodes are communicating with each other.

When an IP datagram is sent to all nodes on a specific network, it is called broadcasting.

There is a third mode of sending an IP datagram called multicasting. In multicasting the IP datagram is delivered to a group of systems identified by a class D address. The systems that have the same multicast address are said to belong to a multicast group. Members of the multicast group, while being assigned a class D address, must also be assigned an IP address (from the class A, B, or C address group). The multicast group can receive an IP datagram in two ways:

- IP datagrams sent directly to their individual IP address (class A, B, C).
- IP datagrams sent to their multicast address (class D).

4.6 CONFIGURING IP ADDRESSES

Hosts on a TCP/IP network need to be configured using proper IP addresses. The actual configuration procedure depends on the operating system. The configuration procedure can be classified into the following categories.

- Command line based
- Menu interface
- Graphical User Interface based
- Obtained dynamically when host boots from a central server

Table 1: Configuring IP Address and other Parameters for Hosts

Method	Operating System/ Protocol
Command line	Unix, VMS, Router, devices, and MS-DOS
Menu interface	Router devices, Unix, NetWare servers, and MS-DOS
Graphical User Interface	Microsoft Windows products.
Dynamically assigned	DHCP and BOOTP protocols. Available on almost all major operating system platforms.

Many systems offer a command that can be executed to modify the IP address. In these systems, the command line is often placed in the startup script for the operating system. The menu interface is built using extended line drawing character sets. You are prompted to enter the IP address and other IP parameters. The Graphical User Interface is for systems — such as X-windows and Microsoft's Windows operating systems—that offer a pixel-based graphical view. The dynamically assigned IP address is used in conjunction with BOOTP or DHP protocol. When starting up, a device requests its IP address and other parameters from a central server that can deliver this information using either the BOOTP or the DHCP protocol.

The IP address information for the host is recorded in a number of places. When the IP address information is entered, it is cached in memory and is available for use by the TCP/ IP software. Alternatively, this information can be recorded on a system file in the operating system. IP addresses of other systems can be discovered by consulting a special file, usually called the "hosts" file, or by using the DNS protocol. The DNS service is typically used to determine a host's IP address given its symbolic DNS name. In some systems, proprietary protocols can be used to discover an IP address on a network. An example of this is the WINS service used in Microsoft's operating systems.

You must consult your operating system manuals for actual details on configuring IP addresses. The following sidebar provides examples of configuring IP addresses for most Unix implementations, and for the Windows NT operating System.

Imagine that you are configuration network interface to IP address 144.19.74.102, subnet mask 255.255.0.0, and directed broadcast address 144.19.255.255.

For a Unix Configuration

1) Log on as root user

2) Run the following command:

Ifconfig eth 0 144.19.74.102 netmask 255.255.0.0. broadcast 144.19.255.255.

Replace eth0 with the Unix logical name of the network interface.

For Windows NT

1) Log on as Administrator user.

2) Select the following:

- Start
- Settings
- Control Panel
- Network
- Select the TCP/IP protocol
- Select Properties

3) You will see a dialog box in which you can enter the IP address and subnet mask.

Windows NT, by default, uses the appropriate directed broadcast address of 144.19.255.255 based on the subnet mask that you specify.

4.7 TCP/IP

Perhaps no other protocols designed to work above the Data Link and Physical OSI layers are as popular as TCP/IP. That's primarily because this global protocol suite has been used by and continually promulgated by thousands of government and educational institutions world-wide.

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) are commonly referred to collectively as TCP/IP. TCP represents a transport layer protocol that provides end to end reliable transmission. To do so, TCP includes such functions as flow control, error control, and exchange of status information. In comparison, IP represents a connectionless-mode network layer protocol designed to route message between networks. In addition to TCP, the Internet suite specifies an optional connection less –mode layer 4 transport protocol known as the User Datagram Protocol(UDP).UDP is used for transaction – based applications, such as transmission of network management information when transmission efficiency is more important than reliability.

Figure 5 illustrates the layering structure of the TCP/IP protocol suite to include a few of the application services included in the suite.

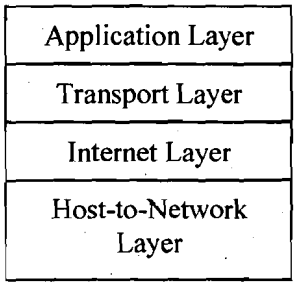


Figure 5: TCP/IP reference model

The advantage of TCP/IP for a network operating system is simple: Interconnectivity is possible for any type of operating system and hardware platform that you might want to add.

TCP/IP is not a single protocol but a set of more than a dozen protocols. Each protocol within the TCP/IP family is dedicated to a different task. All the protocols that make up TCP/IP use the primary components of TCP/IP to send packets of data.

Transmission Control Protocol and Internet Protocol are two of the primary protocols in the TCP/IP family. The different protocols and services that make up the TCP/IP family can be grouped according to their purposes. The groups and their protocols are the following:

Transport: These protocols control the movement of data between two machines.

TCP (Transmission Control Protocol): A connection-based service, meaning that the sending and receiving machines communicate with each other through a stream of messages. TCP has message delivery assurance routines incorporated into it.

☞ **Check Your Progress 1**

1) What is a packet?

.....

.....

.....

2) Which layer of the OSI model has been divided into two sublayers, and what are they?

.....

.....

.....

3) Although UDP is connection less, what is the benefit of UDP?

.....

.....

.....

4) Describe how a TCP message gets from one place to another in step sequence?

.....

.....

.....

4.8 ADDITIONAL TCP/IP-RELATED PROTOCOLS

There are several additional protocols designed to assist TCP and IP. Since routing is so important on a packet-switched network like the Internet, specialized protocols have been designed to assist in this function. Special protocols for determining

addressing on the Internet have also been devised. Additionally, some additional protocols may be involved in error-checking and flow control, just to name a few. Let's explore some of these additional protocols that are included in the TCP/IP suite of protocols.

- **FTP File Transfer Protocol** allows the transfer of copies of files between one node and another. FTP is not hardware-dependent so its services can function just about anywhere. Using this utility to copy data is typically referred to as "FTPing" a file.

- **NFS Network Filing System** was developed by Sun Microsystems Inc. It provides shared access to files in a very transparent and integrated way. This protocol is discussed in more detail a little later.

- **TELNET Remote Terminal Emulation** allows users to communicate with diverse hosts. The TELNET application provides terminal-type access to PCs.

- **UDP User Datagram Protocol** is a bare-bones rapid transmission protocol that uses IP packets to deliver data with no reliability features like connections and ACKs. The forte of UDP is speed, not reliability. It is used in NFS.

- **SMTP Simple Mail Transfer Protocol** is the middleman that uses UDP to move data around from one internetwork host to another. Applications run on both hosts that make use of SMTP.

- **ICMP Internet Control Message Protocol** offers flow control and error-detection to the unreliable delivery method of IP. It provides a facility for routers and gateways on the net to communicate with a source if there is a problem. It also provides a mechanism for determining if a destination cannot be reached.

- **RIP Routing Information Protocol** provides information for routing devices about pathways and number of hops to achieve them. RIP was popularized by its use in a Berkeley UNIX application called "Routed". RIP is ideal for smaller networks, but considered impractical for larger internetworks.

- **ARP & RARP Address Resolution Protocol & Reverse Address Resolution Protocol** are special protocols to allow TCP/IP to interact in environments such as Ethernet. ARP maps TCP/IP addresses to Ethernet Data Link layer addresses. RARP maps the Ethernet Data Link layer address to the TCP/IP address.

That's an overview of some of the better-known additional protocols.

4.9 APPLICATION LAYER PROTOCOLS

The first generation of Internet applications from the late 1960s and early 1970s addressed the need to transfer files between computers, to have remote access to computing resources, and to support communication between users at different hosts. The applications, which met these needs: FTP, Telnet, HTTP, SNMP and DNS, were the predominant applications in the first decades of the Internet. The emergence of the World Wide Web in the early 1990s quickly made web browsing the most popular Internet application. Recently, applications for streaming audio and video, telephony over the Internet, and peer-to-peer file sharing have again changed the landscape of Internet applications.

This section discusses knowledge about how applications interact with the Internet and how applications take advantage of the Internet to provide network services to end-users.

4.9.1 File Transfer Protocol

The **File Transfer Protocol (FTP)** for copying files between computer systems is one of the oldest application- layer protocols and was created before the TCP/IP protocol suite. FTP is a client-server protocol where an FTP client accesses an FTP server. The FTP client authenticates itself with a user name and a password to establish an FTP session. A successful authentication results in the establishment of an FTP session, where the FTP client can download ("get") and upload ("put") files and file lists. When transferring files, FTP respects the ownership and access privileges of files.

Most hosts have a utility program, generally also called FTP, which provides an interactive command line interface to run an FTP session. . FTP clients can also be integrated in other applications, such as web browsers. Anonymous FTP is a special form of file transfer service that allows public access to files at an FTP server. An FTP client can establish an anonymous FTP session by supplying the user name "anonymous" and an arbitrary password (The FTP server usually requests to supply an email address as password).

FTP employs TCP as its transport protocol, thereby ensuring a reliable transfer of transmitted data. Two TCP connections are established for each FTP session, called control connection and data connection. The control connection is used for commands from the client and messages from the server. The data connection is used for the transport of files. FTP server uses the well-known TCP port 21 for the control connection, and the well-known TCP port 20 for the data connection, and an FTP client selects available ephemeral port numbers. The control connection is established at the beginning of the FTP session and stays up for the entire lifetime of the session. The control connection is used by the FTP client for authentication, for setting various session parameters, and for commands to download or upload files. The data connection is opened and closed for each transfer of a file or file list. As soon as a file or a file list has been transferred, the data connection is closed. If there is another file transfer, the data connection is re-opened. By default, the data connection is established upon request by the FTP server. The FTP client starts a TCP server that waits for a connection on an ephemeral port, and sends the port number of this port on the control connection to the FTP server. After the FTP server receives the port, it can request a data connection to the FTP client. A security concern with FTP is that the user name and the password submitted by the FTP client on the control connection at the beginning of an FTP session are not encrypted. Therefore, anyone with the ability to capture traffic from an FTP client can obtain the user name and password used by an FTP client.

The FTP client sends commands to the FTP server, and the FTP server responds with a three-digit response code and an explaining text message. The commands from the FTP client as well as the responses from the FTP server are transmitted as ASCII

Characters.13 The end of a client command and the end of a server response is represented by an end-of-line sequence, which consists of the ASCII special characters Carriage Return (ASCII 10) followed by Line Feed (ASCII 13). When the TCP connection to TCP port 21 of the FTP server is established, the FTP server sends a message that it is ready to interact on a new FTP session. Then, the user supplies a user name and a password. If the authentication is successful, the user sends the IP address and port number of its ephemeral port for the data connection. The IP address and port number is sent in dotted-decimal notation, where the first four numbers

specify the IP address and the last two numbers specify the port number. By default, files are transmitted as text files using ASCII characters. However, the FTP client can change this default so that files are transmitted bit-by-bit without any interpretation. When the file transfer has been completed, the FTP server sends a message to the FTP client. At this 13 ASCII (American Standard Code for Information Interchange) is an encoding format for textual data, which represents an alphanumeric character or special character by seven bits. Application-layer protocols transmit each ASCII character in a byte (octet) with the most significant bit set to zero. The FTP client can download or upload another files, or end the FTP sessions by issuing the command "Quit".

4.9.2 Trivial File Transfer Protocol (TFTP)

The **Trivial File Transfer Protocol (TFTP)** is a minimal protocol for transferring files without authentication and no separation of control information and data as in FTP. Therefore TFTP must not be used on computer where sensitive /confidential information is stored. TFTP is frequently used by devices without permanent storage for copying an initial memory image (bootstrap) from a remote server when the devices are powered on. Due to the lack of any security features, the use of TFTP is generally restricted.

TFTP uses the unreliable transport protocol UDP for data transport, whereas FTP uses TCP. Each TFTP message is carried in a separate UDP datagram. The first two bytes of a TFTP message specify the type of message, which can be a request to download a file, request to upload a file, a data message, or an acknowledgement or error message. A TFTP session is initiated when a TFTP client sends a request to upload or download a file from an ephemeral UDP port to the (well-known) UDP port 69 of a TFTP server. When the request is received the TFTP server picks an ephemeral UDP port of its own and uses this port to communicate with the TFTP client. Thus, both client and server communicate using ephemeral ports.

Since UDP does not recover lost or corrupted data, TFTP is responsible for maintaining the integrity of the data exchange. TFTP transfers data in blocks of 512 bytes. Each block is assigned a 2-byte long sequence number and is transmitted in a separate UDP datagram. A block must be acknowledged before the next block can be sent. When an acknowledgment is not received before a timer expires, the block is retransmitted.

4.9.3 TELNET

Telnet is a remote login protocol for executing commands on a remote host. The Telnet protocol runs in a client-server mode and uses the TCP protocol for data transmission. A client initiates a Telnet session by contacting a Telnet server at a remote host. Recently, the use of Telnet in public networks has been discouraged since Telnet does not offer good protection against third parties that can observe ("snoop") traffic between a Telnet client and a Telnet server. At the Telnet client, a character that is typed on the keyboard is not displayed on the monitor, but, instead, is encoded as an ASCII character and transmitted to a remote Telnet server. At the server, the ASCII character is interpreted as if a user had typed the character on the keyboard of the remote machine. If the keystroke results in any output, this output is encoded as (ASCII) text and sent to the Telnet client, which displays it on its monitor. The output can be just the (echo of the) typed character or it can be the output of a command that was executed at the remote Telnet server.

Telnet uses a single TCP connection for communications. The Telnet server uses the well known TCP port 23 and the Telnet client uses an ephemeral TCP port. After establishing the TCP connection, the Telnet client and server negotiate a set

of parameters for the Telnet session, including terminal type, line speed, if typed characters should be echoed to the client or not, and so on. Unless a Telnet session is explicitly configured not to do so, Telnet client sends one TCP segment for each typed character. Telnet provides some independence from the differences of hardware and software at hosts by mapping input and output to a virtual device, called Network Virtual Terminal (NVT) or pseudo terminal. The Telnet client and Telnet server map the input from a keyboard and the output to a monitor to the ASCII character set. Thus, before a character is sent over the network it is encoded as ASCII character. Also, when an ASCII character is received on the TCP connection, the receiving host interprets the character and translates it into its local character format.

4.9.4 Remote Login

Rlogin is an alternative remote login application program for hosts that run the Unix operating system. Rlogin takes advantage of the fact that both the client and server run a similar operating system, and, for this reason, is simpler than Telnet. Rsh is an application program for the execution of a single command on a remote Unix host. There is also an application program for file transfers between Unix hosts, called rcp. However, this group of applications has poor security features, and is, therefore, often disabled.

The **Secure Shell** suite of protocols provides application layer services for remote login and file transfer services, similar to FTP, Telnet, rlogin, rsh, and rcp, but ensures secure encrypted communications between untrusted hosts. All components of Secure Shell provide authentication, confidentiality, and integrity of data, using a variety of encryption and authentication algorithms, and protect against common attacks on the security or integrity of communications between hosts.

4.9.5 Electronic Mail(Email)

Electronic Mail (email) is the primary Internet service for exchanging messages between users at different hosts(computer) on the Internet. The exchange of email messages is asynchronous, meaning that the transmission and retrieval of an email message can occur at different times.

A user on a host prepares an email on a mail preparation program, called a user agent. Examples of user agents on Unix operating systems are mail, xmail, elm, or pine 14. Email messages are written in plain text and users can add non-text files to an email message. The user agent passes the email to a mail server, where the message is queued for transmission., the user agent and the mail server are on the same host, but it is also possible that they are on different hosts. The mail server uses the application-layer protocol SMTP (Simple Mail Transfer Protocol) to transmit the email message to the mail server of the receiver of the email. The sending mail server uses DNS, the domain name service, to locate the correct remote mail server. In addition to translating host names into IP addresses, DNS also provides the IP addresses of mail servers. For the email ,the mail server at Host A queries DNS for the IP address of the mail server that is responsible for the domain of the email receiver. Once the IP address of the remote mail server is obtained, the sending mail server starts an SMTP client and initiates a TCP connection to the SMTP server of the remote mail server at the well-known TCP port 25. As soon as the TCP connection is established, the SMTP client and server exchange SMTP commands and transfer the email message. These are user agents for hosts with a UNIX operating system. Commands and the email message are transmitted as plain text using ASCII characters. If an email message contains parts that are not text files, these parts are converted to ASCII characters before transmission.. As in FTP, all messages are transmitted as ASCII characters,

using the end-of-line sequence to indicate the end of a command. The SMTP client issues commands to the server, and the server responds to each command with a three-digit response code and explaining text. After the TCP connection is established, the remote mail server sends a brief message.

An email message may traverse multiple SMTP servers before reaching its destination. For example, on most networks a single host is dedicated to handle all outgoing email messages for all hosts of the network. In this case, all hosts forward their outgoing email message to the dedicated mail server, which relays the emails to the proper destinations. Also, in many networks, the receiving mail server does not have access to the receiver's mailbox, and relays incoming messages to the mail server on a host where the receiver's mailbox resides. When a mail server relays an email message, it adds lines to the header of an email message. These lines can be used by the receiver of an email message to trace the path of an incoming email. A mail server adds incoming emails to the mailbox of a user, and assumes that the user has access to the mailbox. Often, however, a user is not permanently connected to its mail server. In such situations, the user can employ mail access protocols to retrieve mail messages from their mailboxes at a remote host. Currently, two popular mail access protocols are in wide use: Post Office Protocol Version 3 (POP3) and Internet Mail Access Protocol (IMAP). Mail access protocols are generally integrated as a component of the user agent.

Exchange of POP3 messages between a POP3 client and a POP3 server. An exchange of POP3 commands between a POP3 client and a POP3 server. All commands and messages are in plain ASCII text, and lines are separated by an end-ofline sequence (CR and LF). The POP3 server acknowledges each command from the client.

For outgoing messages, the user agent at Host X still uses SMTP. , The SMTP client of the user agent at Host X connects to the SMTP server of Host A. Here, Host A acts as a relay and forwards the message to the mail server of the email receiver. Alternatively, the SMTP client at Host X can directly connect to the mail server of the receiver.

Browser-based Emails

Browser based emails allow user to access emails through web browser. The user agent (the web browser) is used HTTP (not POP/IMAP) for the interaction between the browser and email server. When a user wants to send or view the received mail, the browser sends all such commands in the form of HTTP

Example of browser based emails are rediffmail, Gmail, etc.

Multipurpose Internet Mail extensions (MIME)

Traditional email systems are text based. *Multipurpose Internet Mail extensions (MIME)* system extends the basic email system by permitting users to send binary file, e.g., multimedia file, any other arbitrary format

4.10 WORLD WIDE WEB (WWW OR WEB)

The World Wide Web (WWW or Web) emerged in the early 1990s as a new application for access to content stored on Internet hosts. Within a few years, the Web became the most popular Internet application, and traffic on the Internet was dominated by Web applications. The Web is a distributed hypertext system, which is implemented as a client-server application.

A Web client program, called a Web browser, retrieves and displays documents from a Web server. The documents, often referred to as web pages, are formatted using the Hypertext Markup Language (HTML). HTML documents are text files that contain HTML tags, which describe how text should be displayed in the user interface of a Web browser. A hyperlink is special type of tag. It is a reference to another document, which may be located at a different Web server. When displayed in a browser, hyperlinks can be activated with a mouse click.

When activated, the browser retrieves the document that is referenced in the hyperlink. A Web browser makes it convenient to access a variety of documents at different web servers, which refer to each other by hyperlinks, thus, providing users with a feeling of navigating a global database of documents. On the Web, the location of a document is expressed in terms of a Uniform Resource Locator (URL). A URL specifies a unique location for a document on the web. It can reference an HTML document, but also any other file that can be accessed with a Web browser. An example of a URL is http://www.ignou.ac.in/index.html, which specifies that an HTML document with name *index.html* can be accessed via the protocol *HTTP* from a host with the name www.ignou.ac.in

There is no notion of sessions, as, for example, in Telnet and FTP.

In older versions of HTTP, which are still in use today, an HTTP client initiates one TCP connection for each request to the HTTP server. When a single client issues multiple requests to the HTTP server, the number of TCP connections between the HTTP client and HTTP server can grow large. To reduce the number of TCP connections, HTTP/1.1, the current version of HTTP, permits multiple HTTP requests and responses on the same TCP connection, leaves the TCP connection open after a request has been served. The HTTP client does not need to wait until a request is completed before issuing new requests. As in many other application layer protocols of the Internet, HTTP messages are transmitted as ASCII text, using the end-of-line character sequence to indicate the end of a message. The most common HTTP messages sent by a client are requests for HTML or other documents. The server responds to such a request either with a message that contains the requested document or, if the request cannot be satisfied, with an error code. Let us now discuss a request and a response between an HTTP client and an HTTP. Suppose a user has typed the URL is http://www.ignou.ac.in/index.html in a web browser.

When the URL is typed, the web browser starts an HTTP client, which establishes a TCP connection to port 80 of the HTTP server of host www.ignou.ac.in

4.11 DOMAIN NAME SYSTEM

The Internet Protocol address is a 32- bit integer. If somebody wants to send a message it is necessary to include the destination address, but people prefer to assign machines pronounceable, easily remembered names (host names). For this reason the Domain Name System is used. These logical names also allow independence from knowing the physical location of a host. A host may be moved to a different network, while the users continue to use the same logical name. The idea behind domain names is simple: Rather than forcing people to memorize IP numbers, why not give them cryptic names to remember instead?

Domain Name System maps a name to an IP address and conversely an address to a name. Initially when the size of the Internet was small all machines used to maintain a host.txt file, which was passed on, incase of Updates. This host.txt file was centrally managed but looking at the present Internet scenario this does not seem to be a feasible option due to the following reasons:

- Size of the file will be very large, scalability also becomes an issue
- This is centrally managed but since the Internet has distributed management, the management of name space should also be of distributed nature.
- There can be inconsistent results for queries.
- Because of the centralized management as the frequency of lookups increases the time for reply can be very large.

The Domain Name System (DNS) is a distributed database used by TCP/IP applications to map between hostnames and IP addresses, and to provide electronic mail routing information. Each site (university department, campus, company, or department within a company, for example) maintains its own database of information and runs a server program that other systems across the Internet can query. The DNS provides the protocol, which allows clients and servers to communicate with each other.

The system accesses the DNS through a resolver. The resolver gets the hostname and returns the IP address or gets an IP address (*Figure 6*) and looks up a hostname. As we can see in *Figure 6,* the resolver returns the IP address before asking the TCP to open a connection or sending a datagram using UDP.

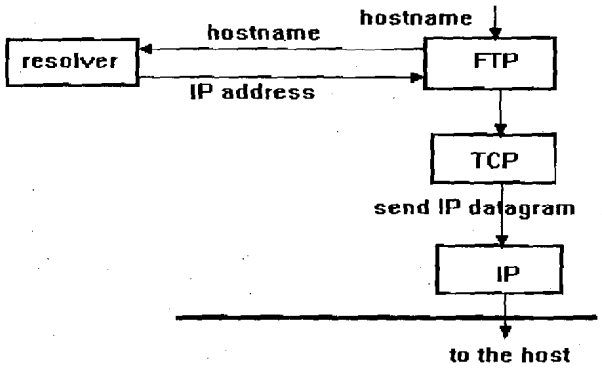


Figure 6: DNS working scheme

DNS Design Goals

The primary goal is a consistent name space, which will be used for referring to resources. Names should not be required to contain network identifiers, addresses, routes, or similar information as part of the name. Name space should be maintained in a distributed manner, with local caching to improve performance. Mechanisms for creating and deleting names; these should also be distributed.

The costs of implementing such a facility dictate that it is generally useful, and not restricted to a single application. We should be able to use names to retrieve host addresses, mailbox data, and other as yet undetermined information. All data associated with a name is tagged with a type, and queries can be limited to a single type.

The name space should be useful in dissimilar networks and applications.

In short design goal of DNS:

1) Distributed ownership: Since the Internet has distributed ownership; the ownership of name space should also be of distributed nature.

2) Have no obvious size limits for names, name components, data associated with a name, etc.

3) DNS protocol should be independent of the network topology.

4) OS/Architecture independent

Design Principles

Hierarchy: The name space as well as management space should be hierarchical. The name space can be represented as a tree with the root label as a null string. The domain name system uses a hierarchical naming scheme known as domain names, which is similar to the Unix file system tree. The root of the DNS tree is a special node with a null label. The name of each node (except root) has to be up to 63 characters. The domain name of any node in the tree is the list of labels, starting at that node, working up to the root, using a period ("dot") to separate the labels (individual sections of a name might represent sites or a group, but the domain system simply calls each section a label). Thus, the domain name "ignou.ac.in" contains three labels: "ignou", "ac", and "in". Any suffix of a label in a domain name is also called a domain. In the above example the lowest level domain is "ignou.ac.in" (the domain name for the IGNOU in India), the second level domain is "ac.in" (the domain name for Academic organizations of India), and the top-level domain (for this name) is "in" (the domain name for India). The node in is the second level node (after root) *(Figure 7)*.

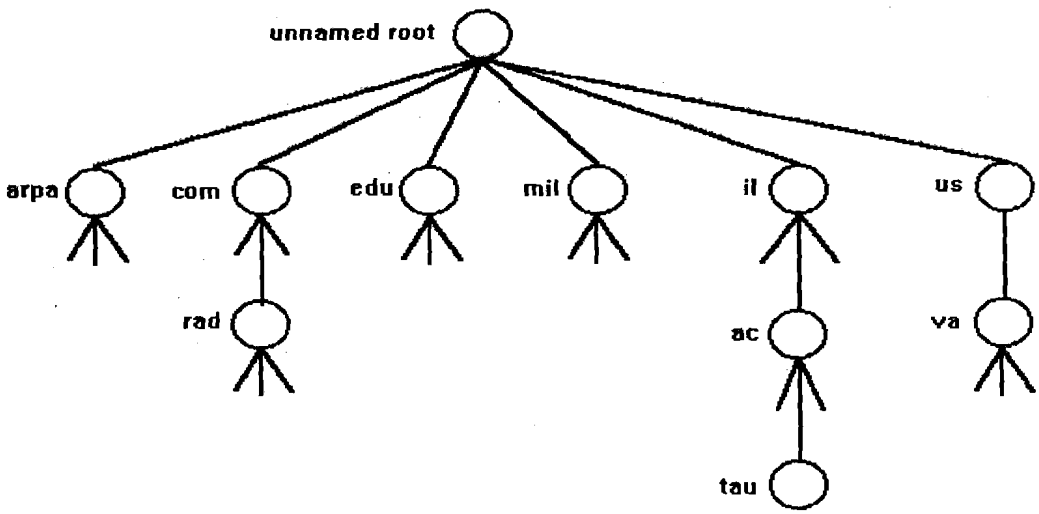


Figure 7: Hierarchical organisation of the DNS

Caching: A fundamental property of the DNS is caching. That is, when a name server receives information about a mapping, it caches that information. Thus a later query for the same mapping can use the cached result. The DNS uses the caching for optimizing search cost. Caching is required as otherwise there will be:

- long time for lookup
- congestion at the root server

Every server has a cache for recently used names as well as a record of where the mapping information for that name was obtained. When a client asks the server to resolve certain name the server does as follows:

1) Check if it has authority for the name. If yes, the server doesn't need caching information.

2) If not, the server checks its cache whether the name has been resolved recently. If yes, the server reports the caching information to its clients.

We can examine the cache when the server cashed the information once, but didn't change it. Since information about a particular name can change, the server may have incorrect information in its caching table. The Time to Live (TTL) value is used to decide when to age information. Whenever an authority responds to a request, it includes a TTL value in the response, which specifies how long it guarantees the binding to remain.

DNS Architecture

NAME SERVERS are server programs, which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree.

REVOLVERS are programs that extract information from name servers in response to client requests. Revolvers must be able to access at least one name server and use that name server's information to answer a query directly.

Data in the DNS consists of Resource Records. There exists a data type for each record. It is of the form (A, MX) where A is the 32-bit IP address, MX is a 16-bit value along with a host name which acts as the mail exchange for the domain. DNS can be used for both forward lookup (host name to IP address) and reverse lookup (IP address to host name). Name space has an entire subtree for reverse mapping.e.g. INADDR.ARPA for reverse lookup

DNS Zones

The zone *(Figure 8)* is a subtree of the DNS that is administered separately. Whenever a new system is installed in a zone, the DNS administrator for the zone allocates a name and an IP address for the new system and enters these into the name server's database. Within a zone DNS service for subsidiary zones may be delegated along with a subsidiary domain. A name server can support multiple zones.

Zones are contiguous regions of the name space, where each can be forked into sub zones. Each of these sub zones can have its independent management.

For example, the IGNOU zone has 2-sub zones cse and lib, which have their own management.

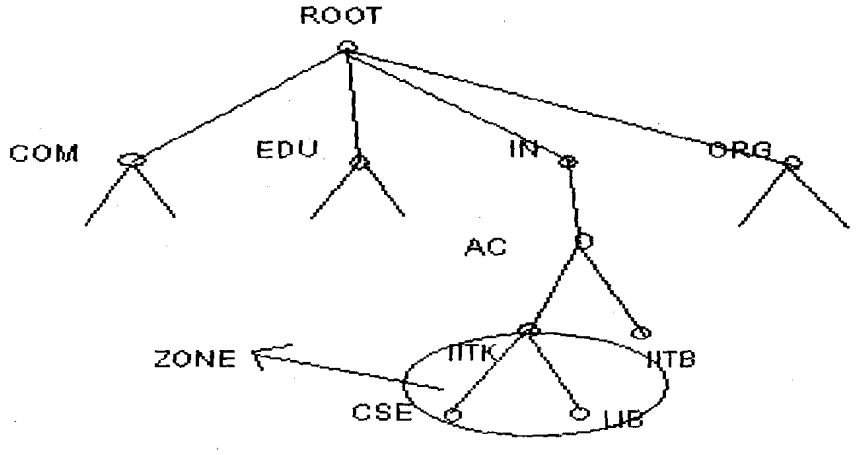


Figure 8: DNS Zones

A name server can support multiple zones; several sub zones can use the same name server. Name servers have pointers among each other.

Remarks on DNS

- DNS uses datagram based access, although a DNS query requires reliability, TCP is not used, as it is a query response mechanism.
- Root server is replicated for improved reliability.
- Address resolution is done recursively. Any DNS server will pass requests it cannot handle to a higher-level server and so on until either the request can be handled or until the root of the DNS name space is reached.

DNS for System break-in

DNS is highly vulnerable to attacks and spoofing. An intruder can intercept virtually all requests to translate names to IP addresses, and supply the address of a subverted machine instead; this would allow the intruder to spy on all traffic, and build a nice collection of passwords if desired.

IP spoofing attacks can be prevented to an extent. Ssh provides an improved type of authentication. The server has a list of host keys stored in /etc/ssh_known_host, and additionally each user has host keys in $HOME/.ssh/known_hosts. Ssh uses the name servers to obtain the canonical name of the client host, looks for its public key in its known host files, and requires the client to prove that it knows the private host key. This prevents IP and routing spoofing attacks.

rlogin and rsh permit ordinary users to extend trust to remote host/user combinations. In that case, individual users, rather than an entire system, may be targeted by source routing attacks. The information required for this attack are the target hostname, trusted hostname and the user name, which are obtained by the "finger" command.

Attack is done as below:

In spoofing a host or application to mimic the actions of another. The attacker pretends to be an innocent host by following IP addresses in network packets. rlogin service can use this method to mimic a TCP connection from another host by guessing TCP sequence numbers.

These attacks can be prevented by:

- Prevent datagram routing with invalid source addresses.
- Introduce unpredictability into connection control mechanisms, such as TCP sequence numbers and the allocation of dynamic port addresses.
- Letting rsh/rlogin to do forward loop along with the reverse lookup.

Allowing to do forward lookup creates a problem called " poisoning the cache " where the attacker sends an unsolicited record along with the PTR record(PTR-a pointer to another part of the domain name space).

This attack can be subverted by rejecting with the record, which arrives along with the PTR record.

4.12 SNMP AND UDP

Simple Network Management Protocol (SNMP)

SNMP *(Figure 9)* is the simple network management protocol. It is used by network management frameworks to manage and monitor network devices, such as hubs and routers. Some computer systems also respond to SNMP queries

SNMP is not actually a protocol: it's a client server application that runs on the UDP (User Datagram Protocol) service of the TCP/IP protocol suite to manage the

network. Network management means to ensure that network is up and running, taking corrective measures and performing maintenance activities. It was developed to be an efficient means of sending network management information over UDP, using Ports 161(SNMP) and 162 (SNMP TRAP).

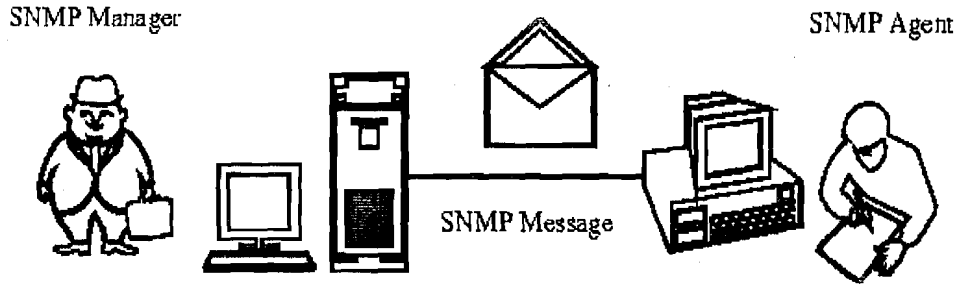


Figure 9: SNMP

Network management system contains two primary elements: a manager and agents. The Manager is the console through which the network administrator performs network management functions. Agents are the entities that interface to the actual device being managed. Bridges, Hubs, Routers or network servers are examples of managed devices that contain managed objects. These managed objects might be hardware, configuration parameters, performance statistics, and so on, that directly relate to the current operation of the device in question. These objects are arranged in what is known as a virtual information database, called a management information base, also called MIB. SNMP allows managers and agents to communicate for the purpose of accessing these objects.

The model of network management architecture looks like *(Figure 10)* this:

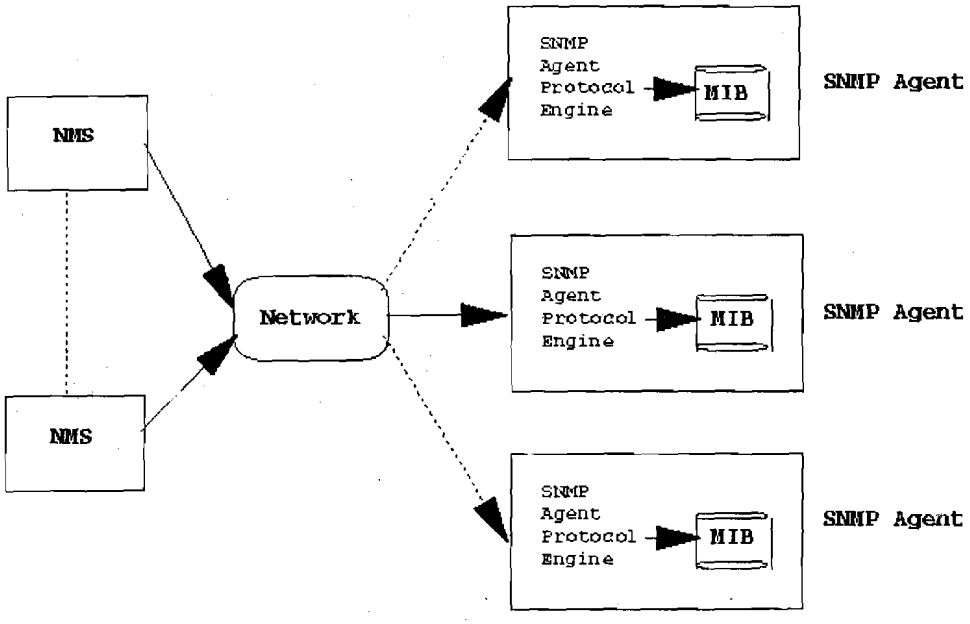


Figure 10: SNMP Architecture

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 variables in agents
  - Acknowledge asynchronous events from agents

SNMP uses the *User Datagram Protocol* (UDP) as the transport protocol for passing data between managers and agents. UDP was chosen over the *Transmission Control Protocol* (TCP) because it is connectionless; that is, no end-to-end connection is made between the agent and the NMS when *datagrams* (packets) are sent back and forth. This aspect of UDP makes it unreliable, since there is no acknowledgment of lost datagrams at the protocol level. It's up to the SNMP application to determine if datagrams are lost and retransmit them if it so desires. This is typically accomplished with a simple timeout. The NMS sends a UDP request to an agent and waits for a response. The length of time the NMS waits depends on how it's configured. If the timeout is reached and the NMS has not heard back from the agent, it assumes the packet was lost and retransmits the request. The number of times the NMS retransmits packets is also configurable.

At least as far as regular information requests are concerned, the unreliable nature of UDP isn't a real problem. At worst, the management station issues a request and never receives a response. For traps, the situation is somewhat different. If an agent sends a trap and the trap never arrives, the NMS has no way of knowing that it was ever sent. The agent doesn't even know that it needs to resend the trap, because the NMS is not required to send a response back to the agent acknowledging receipt of the trap.

The upside to the unreliable nature of UDP is that it requires low overhead, so the impact on your network's performance is reduced. SNMP has been implemented over TCP, but this is more for special-case situations in which someone is developing an agent for a proprietary piece of equipment. In a heavily congested and managed network, SNMP over TCP is a bad idea. It's also worth realizing that TCP isn't magic, and that SNMP is designed for working with networks that are in trouble—if your network never failed, you wouldn't need to monitor it. When a network is failing, a protocol that tries to get the data through but gives up if it can't is almost certainly a better design choice than a protocol that will flood the network with retransmissions in its attempt to achieve reliability.

SNMP uses the UDP port 161 for sending and receiving requests, and port 162 for receiving traps from managed devices. Every device that implements SNMP must use these port numbers as the defaults, but some vendors allow you to change the default ports in the agent's configuration. If these defaults are changed, the NMS must be made aware of the changes so it can query the device on the correct ports.

Figure 11 shows the TCP/IP protocol suite, which is the basis for all TCP/IP communication. Today, any device that wishes to communicate on the Internet (e.g., Windows NT systems, Unix servers, Cisco routers, etc.) must use this protocol suite. This model is often referred to as a protocol stack, since each layer uses the information from the layer directly below it and provides a service to the layer directly above it.

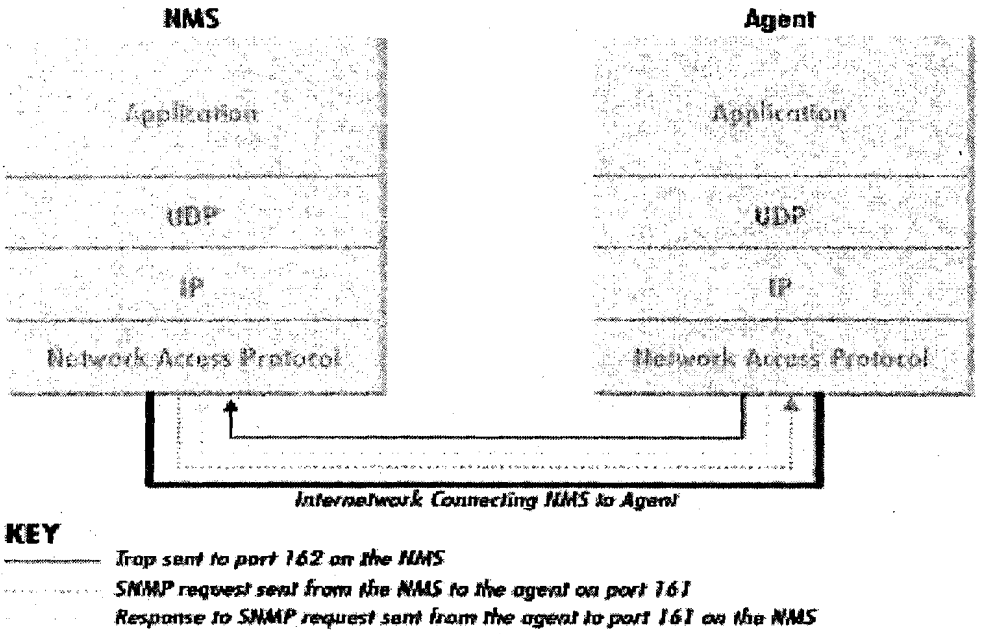


Figure 11: TCP/IP communication model and SNMP

When either an NMS or an agent wishes to perform an SNMP function (e.g., a request or trap), the following events occur in the protocol stack:

Application

First, the actual SNMP application (NMS or agent) decides what it is going to do. For example, it can send an SNMP request to an agent, send a response to an SNMP request (this would be sent from the agent), or send a trap to an NMS. The application layer provides services to an end user, such as operator requesting status information for a port on an Ethernet switch.

UDP

The next layer, UDP, allows two hosts to communicate with one another. The UDP header contains, among other things, the destination port of the device to which it is sending the request or trap. The destination port will either be 161 (query) or 162 (trap).

IP

The IP layer tries to deliver the SNMP packet to its intended destination, as specified by its IP address.

Medium Access Control (MAC)

The final event that must occur for an SNMP packet to reach its destination is for it to be handed off to the physical network, where it can be routed to its final destination. The MAC layer is comprised of the actual hardware and device drivers that put your data onto a physical piece of wire, such as an Ethernet card. The MAC layer also is responsible for receiving packets from the physical network and sending them back up the protocol stack so they can be processed by the application layer (SNMP, in this case).

☞ **Check Your Progress 2**

1) What are the various categories of TCP/IP Well-known Services?

.....

.....

2) What is standard HTTP port?

.....

.....

.....

3) Can a server will work as an FTP server and a webserver?

.....

.....

.....

4) Multiple choice

i) TCP/IP is the main protocol used by computers on the Internet

a) TRUE

b) FALSE

ii) If I wanted to login to a host computer via the Internet, I would use

a) telnet

b) traceroute

c) snmp

d) ftp

iii) To determine if another computer was reachable (alive), the utility I could use is

a) ping

b) nslookup

c) telnet

d) ftp

iv) My computer is very slow. I wish to take advantage of a much faster computer on the network and have my program run there. Which of the following would I use?

a) ftp

b) snmp

c) rsh

d) ping

5) I need to print a document on a printer attached to a UNIX server. Which command would I use?

telnet

b) ping

c) rsh

d) lpr

I wish to gain some statistics from a network server about the number of data packets being sent and received. Which of the following should I use?

a) snmp

b) telnet

c) rsh

d) ftp

4.13 SUMMARY

This unit introduced the building blocks on which internetworks are built. Internetworks are complex systems that, when viewed as a whole, are too much to understand. Only by breaking the network down into the conceptual pieces can it be easily understood. As you read and experience internetworks, try to think of them in terms of OSI layers and conceptual pieces.

Understanding the interaction between various layers and protocols makes designing, configuring, and diagnosing internetworks possible. Without understanding of the building blocks, you cannot understand the interaction between them.

Assigning Internet address to the nodes on the network is a very common task you will perform when building a TCP/IP network. Two types of IP addresses exist: those for IP version 4 and those for IP version 6. IP addresses must be unique on a network. in special cases, it is possible to introduce non-unique addresses, called private addresses.

IP addresses posses a certain structure; they are divided into a network number (netid) and a host number (hostid) portion. This chapter examines the strengths and weaknesses of this scheme. Not all IP addresses can be assigned as a unique identification for network connections. Only class A, B, and C addresses are assignable to individual network connections. Class D is used for IP multicasting. In addition, there are several special addresses used for broadcasting, loop back addresses, and special circumstances. Besides IP & TCP, we looked at several other protocols.

4.14 SOLUTIONS/ANSWERS

Check Your Progress 1

1) A packet is a unit of data that is sent from an originating host to a destination host on a network.

2) Datalink; logical Link Control (LLC) and Media Access Control(MAC)

3) UDP can transfer data faster than TCP

4) (1) First, TCP receives a request from a higher-application that a connection to a remote machine is needed.

   (2) TCP sends a request message to the destination machine. The request message contains a unique number called the socket number that identifies the sending machine's IP address and a port number.

   (3) The message is passed down to IP

   (4) IP assembles a datagram and send it to the destination

(5) The destination's IP, when received the request, sends back its socket number. This process called "exchange of socket number".

(6) TCP sends back up message to the application.

(7) The application then sends data to TCP as a stream of data.

(8) TCP assembles the data in to packets, so called "TCP segments".

(9) TCP then adds a header (Protocol Data Unit) to the data.

(10) TCP send the packaged segment to IP.

(11) IP encapsulates it and sends it over the network, so called "datagram".If there is several segments, a sequence number will created and added to the datagram.

(12) If a segment is missing, TCP sends message back to the sending machine with the faulty sequence number.

(13) After properly received, TCP generated either a positive acknowledgment (ACK), or a request to resend a segment.

Check Your Progress 2

1) The port numbers are divided into three ranges: the Well Known Ports, the Registered Ports, and the Dynamic and/or Private Ports.

   - The well known ports are those from 0 through 1023.
   - The Registered Ports are those from 1024 through 49151
   - The Dynamic and/or Private Ports are those from 49152 through 65535

2) standard port number :80

3) Yes , a server can be running several services and listening at different ports for request

4) 1 (a)

   2 (a)

   3 (a)

   4 (c)

   5 (d)

   6 (a)

4.15 FURTHER READINGS

- *"Internetworking with TCP/IP,* Douglas Comer, Volume I, Fourth Edition", Prentice Hall, 2000.
- *Computer Networking,* Kurose and Ross, Second Edition, Addison-Wesley, 2003 (optional, in RBR).
- *"Computer Networks: A Systems Approach",* Peterson & Davies, Morgan Kaufmann, Second Edition, 2000 W.

- *"TCP/IP Illustrated"*, Volume 1, The Protocols, Richard Stevens, Addison-Wesley, 1994.
- *"Internetworking with TCP/IP"*, Douglas Comer, Volume 3, BSD Socket Version, Prentice Hall, 1993.
- *"Computer Networks"*, Tanenbaum, Third Edition, Prentice-Hall 1996.
- *Data Networks,* Dimitri Bertsekas and Robert Gallager, Second Edition, Prentice-Hall 1992.
- *"OSI A Model for Computer Communication Standards"*, Uyless Black, Prentice Hall, 1991.
- *"Data and Computer Communications"*, William Stallings, Fourth Edition, MacMillan, 1994.

NOTES