NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CLIENT / SERVER COMPUTING

Graduation Project
COM- 400

Student: Omar Awad

Supervisor: Mr. Ümit İlhan

Nicosia - 2004

ACKNOWLEDGEMENTS

First of all, and before saying anything, I wish to thank ***ALLAH*** – *our Creator* – who gave me the morale and opened my mind to get this project done.

Then, I love to send my special thanks to my family, to whom it was not possible to be what I am now without them. I won't ever forget their encouragement and support as long as I am alive. Thank you father, you have always been my ideal. Mother, I am so glad to thank you at this particular moment of my life, your prayers helped making this day come true. I also thank Fady, you are my big brother, and I guess this word explains my feelings. Finally special thanks to my little sister: Fadia. I love you all so much.

After that, I would like to show my special appreciations to my supervisor *Mr. Umit Ilhan* for his kindness, humility, patience, and support. He was actually very helpful, and provided me with brotherhood, which I – a foreign student – miss in this far away country from my homeland.

And I hope to thank all my friends, specially my best friends *Fady El-Lada'a & Allam Hijjawy* for helping, supporting, and guiding me through my study. Finally, thanks to everybody, and may *Allah* bless you all.

ABSTRACT

TCP and IP were developed by a Department of Defense (DOD) research project to connect a number of different networks designed by different vendors into a network of networks (the "Internet"). It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN.

The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet. On the battlefield a communications network will sustain damage, so the DOD designed TCP/IP to be robust and automatically recover from any node or phone line failure. This design allows the construction of very large networks with less central management. However, because of the automatic recovery, network problems can go undiagnosed and uncorrected for long periods of time. There are dramatic improvements in the hardware and software technologies for microcomputers. Microcomputers become affordable for small businesses and organizations. And at the same time their performances are becoming more and more reliable.

TCP/IP is made up of two acronyms, TCP, for Transmission Control Protocol, and IP, for Internet Protocol. TCP handles packet flow between systems and IP handles the routing of packets. However, that is a simplistic answer that we will expound on further.

All modern networks are now designed using a layered approach. Each layer presents a predefined interface to the layer above it. By doing so, a modular design can be developed so as to minimize problems in the development of new applications or in adding new interfaces.

TABLE OF CONTENTS

INTRODUCTION

The most common networks are Local Area Networks or LANs for short. A LAN connects computers within a single geographical location, such as one office building, office suite, or home. By contrast, Wide Area Networks (WANs) span different cities or even countries, using phone lines or satellite links. The success of a network will depend on the people involved and their commitment to the network and the competitive production of high quality pork. Networks that are based on sound technology to generate real value will also be more successful.

Client-Server Computing is divided into three components, a Client Process requesting service and a Server Process providing the requested service, with a Middleware in between them for their interaction.
The OSI Reference Model is composed of seven layers. TCP/IP was designed with layers as well, although they do not correspond one-to-one with the OSI-RM layers. You can overlay the TCP/IP programs on this model to give you an idea of where all the TCP/IP layers reside.

So you've just been told you are on a TCP/IP network, you are the new TCP/IP system administrator, or you have to install a TCP/IP system. But you don't know very much about TCP/IP. That's where this project comes in. You don't need any programming skills, and familiarity with operating systems is assumed. Even if you've never touched a computer before, you should be able to follow the material.

By the time you finish this project, you will understand the different components of a TCP/IP system, as well as the complex acronym-heavy jargon used. Following the examples presented, you should be able to install and configure a complete TCP/IP network for any operating system and hardware platform.

CHAPTER ONE
NETWORKING OF COMPUTER

1.1 Overview

Networks are collections of computers, software, and hardware that are all connected to help their users work together. Also computer networks inclode the desire to (Access, Save, Share, and backup data files in a central location), (Share prinphral devices including : Printers, Scanners, Fax machines, and Modems), (Control access to sensitive information).

A network connects computers by means of cabling systems, specialized software, and devices that manage data traffic. Computer networks fall into two main types: *client/server networks* and *peer-to-peer networks*. A client/server network uses one or more dedicated machines (the server) to share the files, printers, and applications. A peer-to-peer network allows any user to share files with any other user and doesn't require a central, dedicated server.

The most common networks are Local Area Networks or LANs for short. A LAN connects computers within a single geographical location, such as one office building, office suite, or home. By contrast, Wide Area Networks (WANs) span different cities or even countries, using phone lines or satellite links.

1.2 Short History of Computer Networking

The field of computer networking and today's Internet trace their beginnings back to the early 1960s, a time at which the telephone network was the world's dominant communication network. That the telephone network uses circuit-switching to transmit information from a sender to receiver - an appropriate choice given that voice is transmitted at a constant rate between sender and receiver. Given the increasing importance (and great expense) of computers in the early 1960's and the advent of timeshared computers, it was perhaps natural (at least with perfect hindsight!) to consider the question of how to hook computers together so that they could be shared among geographically distributed users. The traffic generated by such users was likely to be "bursty" - intervals of activity, e.g., the sending of a command to a remote computer, followed by periods of inactivity, while waiting for a reply or while contemplating the received response.

1.3 What Is Networking of Computers?

A network consists of two or more computers that are linked in order to share resources (such as printers and CD-ROMs), exchange files, or allow electronic communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams.

1.4 The Basic Types of Networks Include

* *Local Area Network (LAN)*
* *Metropolitan Area Network (MAN)*
* *Wide Area Network (WAN)*

1.4.1 Local Area Network

A Local Area Network (LAN) is a network that is confined to a relatively small area. It is generally limited to a geographic area such as a writing lab, school, or building. Rarely are LAN computers more than a mile apart.

In a typical LAN configuration, one computer is designated as the file server. It stores all of the software that controls the network, as well as the software that can be shared by the computers attached to the network. Computers connected to the file server are called workstations. The workstations can be less powerful than the file server, and they may have additional software on their hard drives. On most LANs, cables are used to connect the network interface cards in each computer.

1.4.2 Metropolitan Area Network

A Metropolitan Area Network (MAN) covers larger geographic areas, such as cities or school districts. By interconnecting smaller networks within a large geographic area, information is easily disseminated throughout the network. Local libraries and government agencies often use a MAN to connect to citizens and private industries. One example of a MAN is the MIND Network located in Pasco County, Florida. It connects all of Pasco's media centers to a centralized mainframe at the district office by using dedicated phone lines, coaxial cabling, and wireless communications providers.

1.4.3 Wide Area Network

Wide Area Networks (WANs) connect larger geographic areas, such as Florida, the United States, or the world. Dedicated transoceanic cabling or satellite uplinks may be used to connect this type of network.

Using a WAN, schools in Florida can communicate with places like Tokyo in a matter of minutes, without paying enormous phone bills. A WAN is complicated. It uses multiplexers to connect local and metropolitan networks to global communications networks like the Internet. To users, however, a WAN will not appear to be much different than a LAN or a MAN.

1.5 Benefits of Networking

Below are some major benefits from networking. However, you may discover other benefits not listed here.

1.5.1 Capture Proven Technology

One of the greatest benefits of networking is that it allows producers to use technology that cannot be used individually. For example, multi-site production has been shown to improve feed efficiency and average daily gain by improving the health status of the animal. This technology is difficult and often costly to use within a single operation, but lends itself well to networking with other producers. One person may farrow, a second has the nursery, and the third finishes the pigs. In addition to improved herd health, each operation is able to specialize in one aspect of production. Networks built on scientific information and technology that produce real returns have a better chance of offsetting their costs. While not all technologies lend themselves to networking, many do if producers are objective, seek professional advice, and are committed to making them work.

1.5.2 Capture Real Economies

Like technology, producers may also be able to generate cost savings by networking with other producers to capture economies in volume sales and purchases. Cooperative genetic multipliers are an example where producers have worked together to produce their own gilts, are able to get state-of-the-art genetics at wholesale prices, and reduce their genetic access cost by $4 to $6 per pig. Volume purchases of a

manufactured product can reduce costs. However, make sure the savings are worth the effort of networking. Focus on the big ticket items such as feed and labor rather than the less important factors of production. For example, saving 30 percent on the price of vaccines and antibiotics when total vet and medical cost represent only 3.6 percent of total cost is insignificant. Conversely, cutting feed cost per pound of gain by 30 percent will make a real difference. However, the way to save 30 percent on feed cost is through improved efficiency using proven technologies, because there typically is not that much savings available on purchase price.

1.6 Evaluating Networking

Before weighing the benefits and limitations of joining a network, the producer should research the particular network, then evaluate the network on its own merit. A few key questions to ask include:

- What are the mission and objectives of the network?
- Who are the members of the network?
- Do all members have the same objectives?
- Who is the leader / champion?
- Will it lower my cost of production?
- Will it improve my market access?
- Does the network fit my mission and goals?
- Is the network positioned to evolve?
- What is the complete impact on my operation?

When evaluating a networking opportunity keep two old sayings in mind: If it sounds too good to be true it probably is, and there's no such thing as a free lunch.

1.7 Limitations Of Networking

Below are some of the major limitations of networking. However, you should be alert to other limitations that may arise in your situation.

1.7.1 Commitment of People

If networking was easy, everyone would be doing it. Thus, networking should not be viewed as a magic cure-all as it requires work, leadership, and commitment. Networking will seldom make a poor manager better, but it may allow him or her the

opportunity to improve. Because pork production has been relatively profitable for quite some time, it is often difficult for producers to see the need for change or agree on what that change should look like. Many producers will have trouble accepting a compromise if they have to give up something even if it is a win-win solution. Networks also require long term commitment that may require producers to forgo some short term gains. Pooled marketing is a classic example. Pooling hogs to develop market leverage creates a shortage of hogs in the local market and prices may be bid above the pool, price to attract hogs. If producers are not committed to the network and sell hogs outside of the pool they have lost the long term benefit from group marketing. Thus, the success of a network depends on the efforts of the leaders and the commitment of all members.

1.7.2 Joint Responsibility

Networking typically implies that the success of the group depends on the performance of each individual member. Some producers may not want the responsibility of determining their fellow member's future, along with their own. Others may not trust their neighbor to deliver the goods. For example, the pressure may be on the marketing negotiator to extract the best price bid for everyone; or it may be the responsibility of the member who is breeding and farrowing sows to keep new finishing buildings full for fellow members. Often these agreements can be formalized through checks and balances or contracts, but they require increased communication, business structure, and overhead.

1.7.3 Loss of Control

Along with sharing responsibility, networking also requires that members turn over some degree of control. Loss of control may be as simple as complying with a marketing date, or it may require changing input suppliers, management practices, facilities, and priorities. If a member has properly researched the network, has been involved in its development, or has accepted the mission of the network, he should accept the results. However, human nature often shortens memories and clouds logic. Some networks require an investment of capital. Others require that management be turned over to someone else. For some producers, the cost of following another person's management requirements may offset any benefits from improved production efficiency.

1.7.4 Formal Business Procedures

Although many networks are informally structured and loosely operated, tighter control and more formal business procedures are necessary as transactions become more complicated. Networking requires increased communication because more than one individual is involved. The minimum cost of communication is the higher overhead cost of the time involved in keeping people informed. However, a higher cost may result if communication breaks down and members feel that they are not being kept informed. Effective communication may involve newsletters, meetings, memos, etc. Networking typically involves financial transactions, and credit-worthiness and collection procedures become essential. Formalizing a network to include articles of incorporation, bonding, licensing, etc., is often costly, but is much less expensive than trying to correct a problem later.

1.7.5 Loss of Markets and Suppliers

Networking typically involves direct negotiation and contract agreements with a supplier or buyer. As a result, the product or commodity is removed from the open market. At a minimum, a physical market such as a terminal market or buying station may close. In a broader sense, the market is less actively traded, no longer represents the direct trade product, and may disrupt the price discovery process. Problems arise if the network negotiates a price based on a certain market (i.e., the Omaha terminal price) and that market closes or its trades are based on inferior products. While some argue that networks are a consequence of thin markets, they may also accelerate a change in markets. Just as local open markets may close as producers enter selling networks, input suppliers may close as purchasing networks buy wholesale to bypass retail vendors. Local suppliers include lenders, veterinarians, equipment and supply stores, as well as feed dealers. The impact on these businesses will also be felt elsewhere on main street. Also in some cases the local supplier may be included in the network or negotiate for the network's business. However, even if local businesses are in the network, there may be one or more that are negatively affected. Also, remember that successful networks will be those that capture real economies. It may not be worth the potential damage to the local economy to save one percent on cost by purchasing inputs cheaper if improved efficiency due to adopting appropriate technology can lower cost of production by 20 percent.

1.8 Types of Networks

It can also talk to other machines via a variety of different networks. The most common types of network depending on scale are normally described as **LAN**s (Local Area Networks) or **WAN**s (Wide Area Networks). The Internet is merely a world wide collection of these.

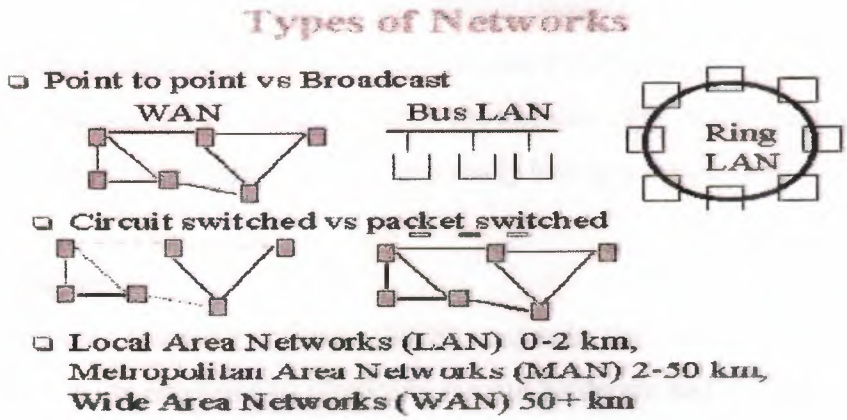


Figure 1.1 Types of Networking

1.9 Networking Technology Consists of Hardware and Software Components.

The primary hardware components are the NIC (network interface card) and the wiring that connects the population of the network to each other. The NOS, (Network Operating System), is the software component that enables communications over the network's hardware. At the most basic level, anything you can physically touch is hardware (i.e. a computer, a floppy disk, cables, printers, circuit boards.) Software on the other hand, is a little more intangible. Software will always live on hardware but you never touch it directly. For example, the floppy disk, hard drive, or backup tape you might have is **hardware**; the program/data stored on the floppy disk, hard drive, or backup tape is **software**.

1.9.1 Hardware

The network interface card (NIC) is typically a circuit board installed in a computer. There are also models that can be attached externally like the XIRCOM pocket ethernet adapters. (XIRCOM is a reliable brand name, but there are of course other manufacturers of possibly equal quality). An external network card will either attach to a serial or parallel port of the computer. Wiring, or network cabling, physically links the nodes or components of the network to each other. Network cabling will

normally follow one of two wiring topology schemes - a star network topology or a linear-bus topology. Each topology has its strengths. A star network has great advantages over a linear-bus network in terms of fault tolerance and modularity. Many small offices get a quick start in networking by installing a linear-bus topology network. This is largely due to the fact that the network wiring can be laid across the floor, behind desks and cabinets, making installation cheaper and permission from landlords to run wiring unnecessary. As a network grows, it often changes to a star topology - as expanding the linear-bus wiring becomes problematic. The star topology is much more common than the linear-bus topology, especially in a new installation.

1.9.2 Software

A network operating system, (NOS), expands the resources of a computer by creating virtual connections to physically remote hardware and software. For example, a user may sit at his/her desk and be able to print to a printer, photocopier, or fax that they have no direct physical connection to. Likewise, the same user may have access to accounting software that lives on a fileserver, (read fileserver as centrally shared computer), many feet or even miles away from their desk. A NOS accomplishes this by convincing the computer it has a direct connection to these remote resources. The computer is convinced by the creation of 'mappings' that establish a virtual connection to the remote device, mimicking a physical connection to the device. Potentially, a networked user will have access to every printer or peripheral device on a network and every megabyte of central storage space on the fileservers. For example, a user on a Novell Netware network will in addition to their 'local' A: and C: drives have an F:, G:, H:, etc... drive. These 'remote' drives (F:, G:, H:, etc.) will be virtual drive mappings to remote resources. Likewise their printer ports are not limited to local hardware and may be 'mapped' to network printing devices.

1.10 What is a Network Operating System?

Unlike operating systems, such as DOS and Windows, that are designed for single users to control one computer, network operating systems (NOS) coordinate the activities of multiple computers across a network. The network operating system acts as a director to keep the network running smoothly.

The two major types of network operating systems are:

* *Peer-to-Peer*
* *Client/Server*

1.10.1 Peer-to-Peer

Peer-to-peer network operating systems allow users to share resources and files located on their computers and to access shared resources found on other computers. However, they do not have a file server or a centralized management source. In a peer-to-peer network, all computers are considered equal; they all have the same abilities to use the resources available on the network. Peer-to-peer networks are designed primarily for small to medium local area networks. AppleShare and Windows for Workgroups are examples of programs that can function as peer-to-peer network operating systems.

Figure 1.2 Peer-To-Peer Network

1.10.1.1 Advantages of A Peer-To-Peer Network:

1- Less initial expense - No need for a dedicated server.

2- Setup - An operating system (such as Windows XP) already in place may only need to be reconfigured for peer-to-peer operations.

1.10.1.2 Disadvantages of A Peer-To-Peer Network:

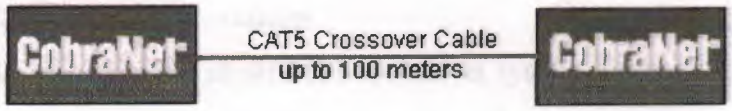
1- Decentralized - No central repository for files and applications.

2- Security - Does not provide the security available on a client/server network.

Note: *The second Type of Network (Client-Server) It Will Be Explan In the Second Chapter of The Project.*

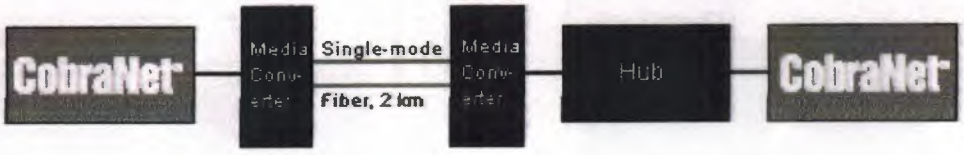
1.11 Exampels of Networking

The following figures show some examples of typical CobraNet designs for repeater networks (i.e., a network utilizing only repeater hubs - no switches). When you've finished reviewing the examples, be sure to perform the compliancy test to determine if the CobraNet repeater network you've designed is CobraNet Simple Configurations Using Category 5 cable, the maximum distance between two CobraNet devices is 100 meters.



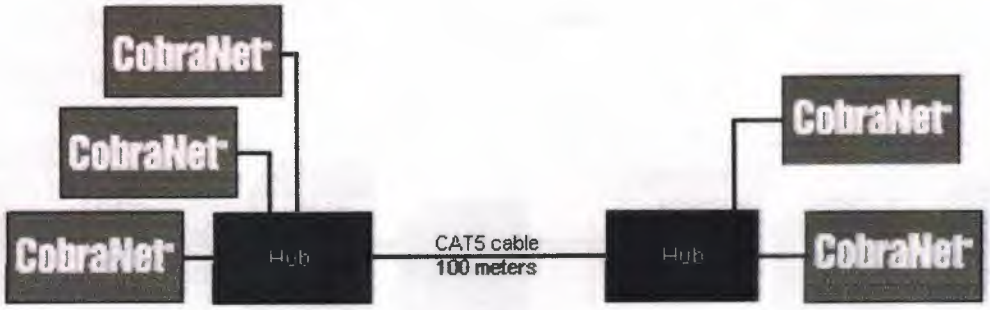
To extend the distance between devices slightly, add a hub. Now the maximum distance between devices is increased to 200 meters...

If longer runs are required, fiber optic cable may be used. In its simplest point-to-point form, a CobraNet device can connect to another device located a maximum of 2 kilometers away. To accomplish this, simply connect the network devices to media converters. A hub must be present between one of the CobraNet devices and media converters for proper operation



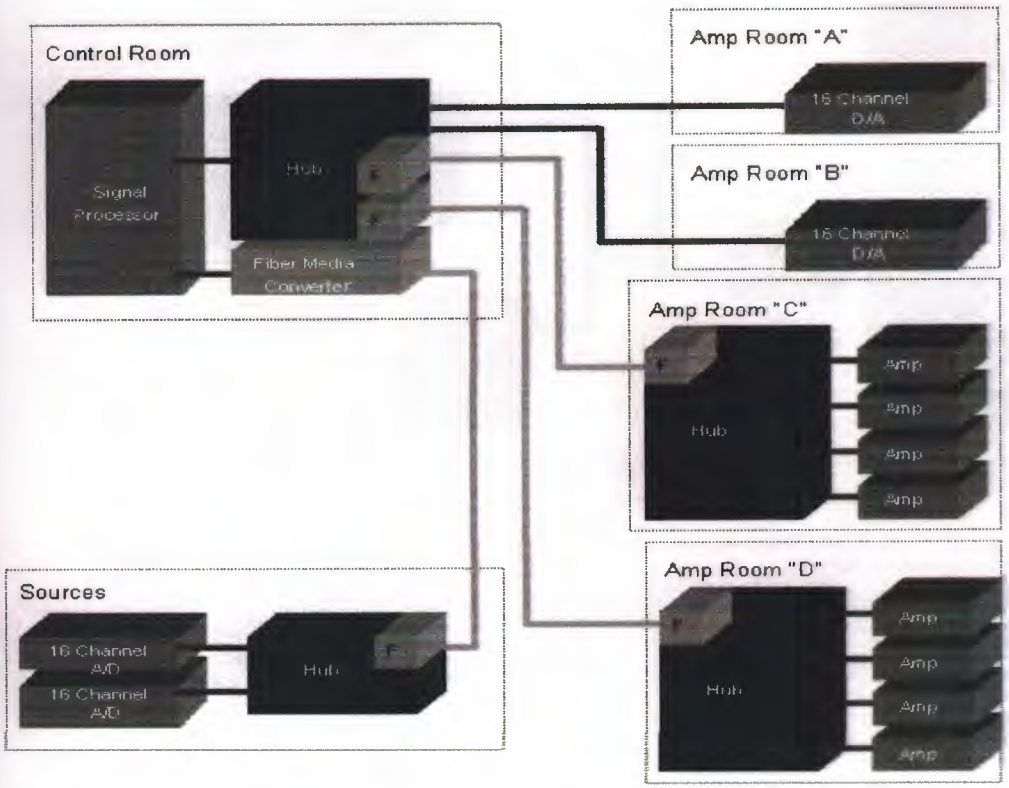
The real advantage of using a hub in a CobraNet network is not to extend the distance between devices, but to allow multiple CobraNet devices to connect to the network. The

example below utilizes Category 5 cable to connect eight CobraNet devices to the network. If the distance between the hubs is more than 100 meters, simply add media converters at both ends and use fiber.



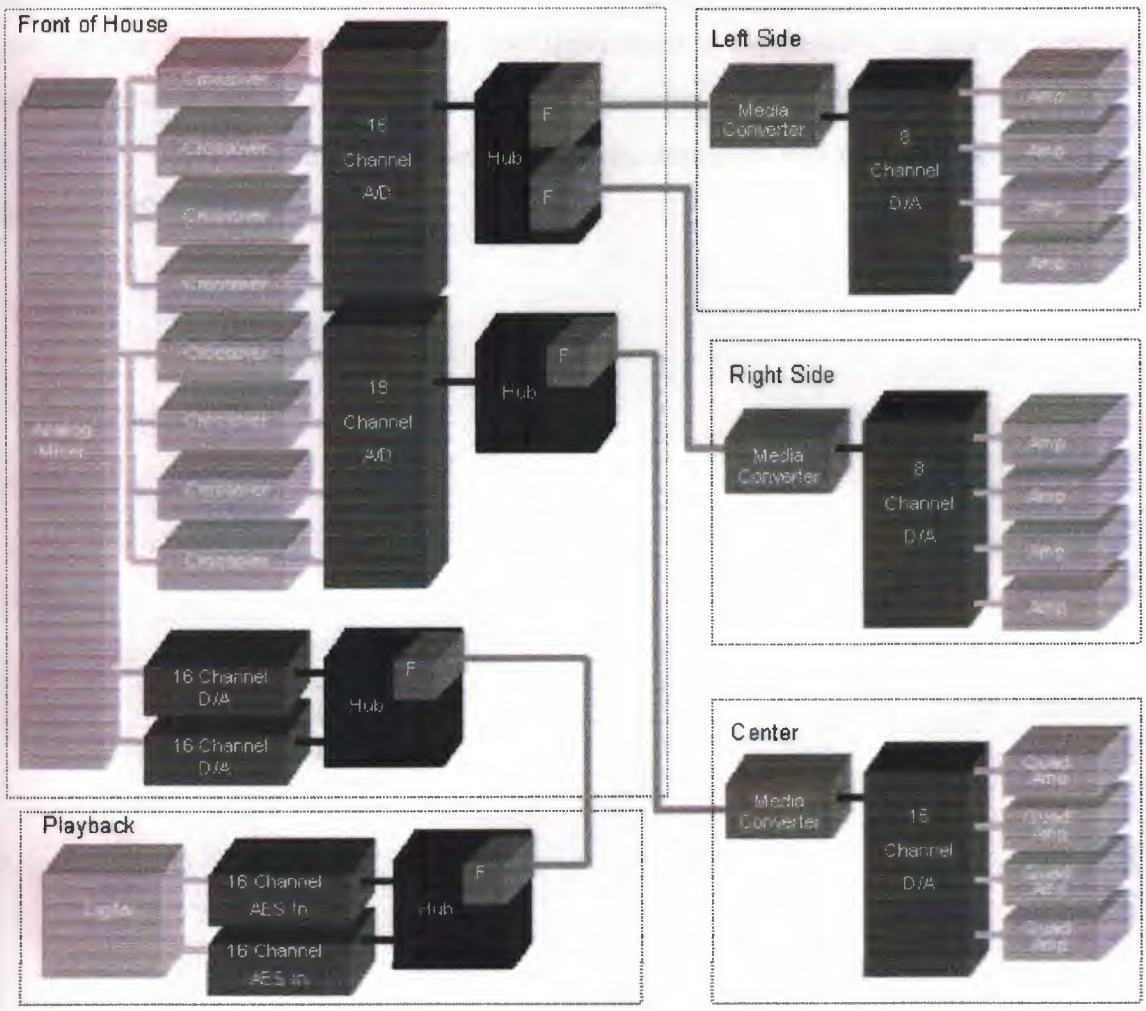
1.11.1 More Complex Configurations

The following example illustrates a CobraNet system for a stadium with multiple equipment rooms. All audio sources are sent via fiber to the main signal processor located in a remote control room. Here, the signals are processed and sent to another network which delivers the audio signals to Amp Rooms "A", "B", "C" and "D".



1.11.2 Half-time Show Example

The following is an example of a live sound event system, similar to the system used at two recent Super Bowl half-time shows. Each of the A/D and D/A boxes are CobraNet devices. A remote truck supplies 32 channels of playback through a fiber to the front of house mixer. The mix is sent to crossovers that feed another network to three remote amplifier locations.



1.12 Summary

In summary, networking provides progressive producers the opportunity to keep up with a changing industry. It should not be viewed as a way to avoid change or as a last ditch effort to stay afloat. The success of a network will depend on the people involved and their commitment to the network and the competitive production of high quality pork. Networks that are based on sound technology to generate real value will also be more successful. Working both horizontally and vertically in the pork channel is not without its costs. Members may find themselves in the position of picking winners and losers and may at times forgo short term gains to achieve long run sustainable advantages. Also, as with any leader in business, members will need to take risks and may be criticized by peers.

CHAPTER TWO
CLIENT-SERVER COMPUTING & FUNDAMENTALS

2.1 Overview

The evolution of Client-Server Computing has been driven by business needs, as well as the increasing costs for host (mainframe and midrange) machines and maintenance, the decreasing costs and increasing power of microcomputers and the increased reliability of LANs (Local Area Networks).

In the past twenty years, there are dramatic improvements in the hardware and software technologies for microcomputers. Microcomputers become affordable for small businesses and organizations. And at the same time their performances are becoming more and more reliable.

On the other hand, the drop in price for mainframe is growing at a slower rate than the drop in its price. Little developments have achieved with mainframes.

The following are the improvements made by microcomputers:

- **Hardware:** The speed of desktop microprocessors has grown exponentially, from 8MHz 386-based computers to 100Hz-based Pentium-based microprocessors. These mass-produced microprocessors are cheaper and more powerful than those used in mainframe and midrange computers. On the other hand, the capacity of main memory in microcomputers has been quadrupling every three years. Typically main memory size is 16 Megabytes nowadays. Besides, the amount of backup storage and memory such as hard disks and CD-ROMs that are able to support microcomputers has also puts an almost unlimited amount of data in reach for end-users.
- **Software:** The development and acceptance of GUIs (Graphical User Interfaces) such as Windows 3.1 and OS/2 has made the PC working environment user-friendlier. And the user is more efficient in learning new application software in a graphical environment. Besides GUIs, the use of multithreaded processing and relational databases has also contributed to the popularity of Client-Server Computing.

2.2 Configurations in Client-Server Computing

Client-Server Computing is divided into three components, a Client Process requesting service and a Server Process providing the requested service, with a Middleware in between them for their interaction.

2.2.1 Client

A Client Machine usually manage the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs. It is the front-end of the application that the user sees and interacts with. Besides, the Client Process also manages the local resources that the user interacts with such as the monitor, keyboard, workstation, CPU and other peripherals.

The client is a process (program) that sends a message to a server process (program), requesting that the server perform a task (service). Client programs usually manage the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs, and sometimes execute business logic. The client-based process is the front- end of the application that the user sees and interacts with.

The client process contains solution-specific logic and provides the interface between the user and the rest of the application system. The client process also manages the local resources that the user interacts with such as the monitor, keyboard, workstation CPU and peripherals. One of the key elements of a client workstation is the graphical user interface (GUI). Normally a part of operating system i.e. the window manager detects user actions, manages the windows on the display and displays the data in the windows.

2.2.2 Server

On the other side, the Server Machine fulfills the client request by performing the service requested. After the server receives requests from clients, it executes database retrieval , updates and manages data integrity and dispatches responses to client requests. The server-based process may run on another machine on the network; the server is then provided both file system services and application services. Or in

some cases, another desktop machine provides the application services. The server acts as software engine that manages shared resources such as databases, printers, communication links, or high powered-processors. The main aim of the Server Process is to perform the back-end tasks that are common to similar applications.

The simplest form of servers are disk servers and file servers. With a file server, the client passes requests for files or file records over a network to the file server. This form of data service requires large bandwidth and can slow a network with many users. The more advanced form of servers are Database servers, Transaction server and Application servers.

The following diagram illustrates the relationship between client and server computers. The client requests information; the server processes the request and sends a response back to the client.

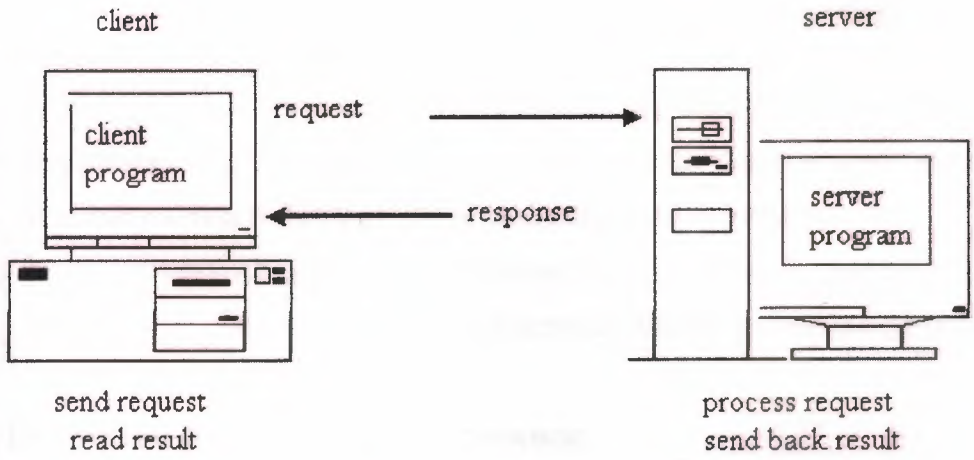


Figure 2.1 The Relationship between Client and Server Computers

2.3 What Is Client/Server Method ?

Client/server network operating systems allow the network to centralize functions and applications in one or more dedicated file servers . The file servers become the heart of the system, providing access to resources and providing security. Individual workstations (clients) have access to the resources available on the file

servers. The network operating system provides the mechanism to integrate all the components of the network and allow multiple users to simultaneously share the same resources irrespective of physical location. Novell Netware and Windows NT Server are examples of client/server network operating systems.

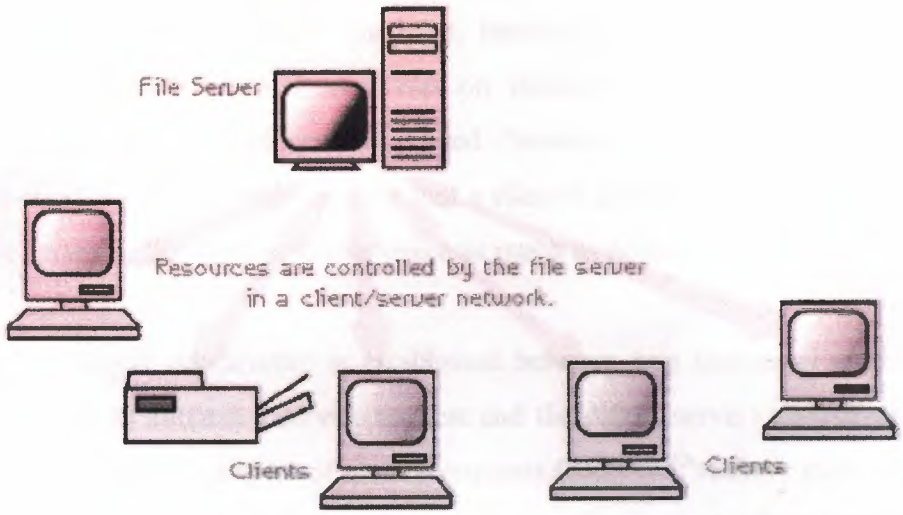


Figure 2.2 Client/Server Network

2.3.1 Advantages Of a Client/Server Network:

1. Centralized - Resources and data security are controlled through the server.
2. Scalability - Any or all elements can be replaced individually as needs increase.
3. Flexibility - New technology can be easily integrated into system.
4. Interoperability-All components (client/network/server) work together.
5. Accessibility - Server can be accessed remotely and across multiple platforms.

2.3.2 Disadvantages of a Client/Server Network:

1. Expense - Requires initial investment in dedicated server.
2. Maintenance - Large networks will require a staff to ensure efficient operation.
3. Dependence - When server goes down, operations will cease across the network.

2.4 Client/Server Fundamentals

2.4.1 Definitions

Client/server model is a concept for describing communications between computing processes that are classified as service consumers (clients) and service

providers (servers). Figure 2.3 presents a simple C/S model. The basic features of a C/S model are:

1. Clients and servers are functional modules with well defined interfaces (i.e., they hide internal information). The functions performed by a client and a server can be implemented by a set of software modules, hardware components, or a combination thereof. Clients and/or servers may run on dedicated machines, if needed. It is unfortunate that some machines are called "servers." This causes confusion (try explaining to an already bewildered user that a client's software is running on a machine called "the server"). We will avoid this usage as much as possible.

2. Each client/server relationship is established between two functional modules when one module (client) initiates a service request and the other (server) chooses to respond to the service request. Examples of service requests (SRs) are "retrieve customer name," "produce net income in last year," etc. For a given service request, clients and servers do not reverse roles (i.e., a client stays a client and a server stays a server). However, a server for SR R1 may become a client for SR R2 when it issues requests to another server (see Figure 2.3). For example, a client may issue an SR that may generate other SRs.

3. Information exchange between clients and servers is strictly through messages (i.e., no information is exchanged through global variables). The service request and additional information is placed into a message that is sent to the server. The server's response is similarly another message that is sent back to the client. This is an extremely crucial feature of C/S model.

The following additional features, although not required, are typical of a client/server model:

4. Messages exchanged are typically interactive. In other words, C/S model does not support an off-line process. There are a few exceptions. For example, message queuing systems allow clients to store messages on a queue to be picked up asynchronously by the servers at a later stage.

5. Clients and servers typically reside on separate machines connected through a network. Conceptually, clients and servers may run on the same machine or on separate machines. In this Project, however, our primary interest is in *distributed client/server systems* where clients and servers reside on separate machines.

The implication of the last two features is that C/S service requests are real-time messages that are exchanged through network services. This feature increases the appeal of the C/S model (i.e., flexibility, scalability) but introduces several technical issues such as portability, interoperability, security, and performance.

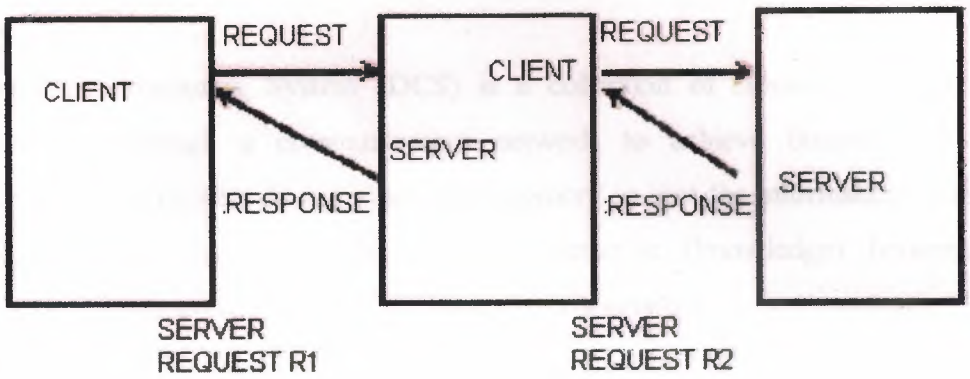


Figure 2.3 Conceptual Client/Server Model

2.4.2 Client/Server–A Special Case of Distributed Computing

Figure 2.4 shows the interrelationships between distributed computing and client/server models. Conceptually, client/server model is a special case of distributed-computing model.

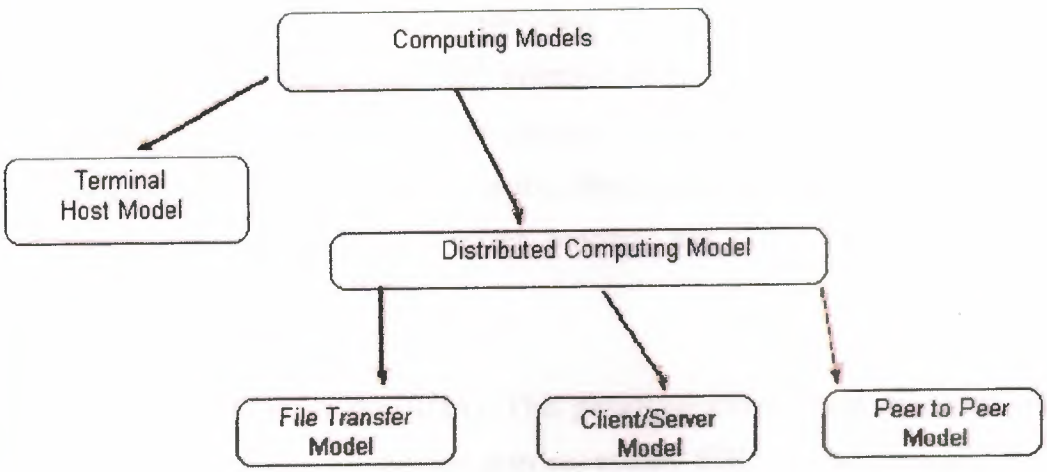


Figure 2.4 Interrelationships between Computing Models

A Distributed Computing System (DCS) is a collection of autonomous computers interconnected through a communication network to achieve business functions. Technically, the computers do not share main memory so that the information cannot be transferred through global variables. The information (knowledge) between the computers is exchanged only through messages over a network.

The restriction of no shared memory and information exchange through messages is of key importance because it distinguishes between DCS and shared memory multiprocessor computing systems. This definition requires that the DCS computers are connected through a network that is responsible for the information exchange between computers. The definition also requires that the computers have to work together and cooperate with each other to satisfy enterprise needs.

2.5 Client/Server Architectures

Client/server architecture provides the fundamental framework that allows many technologies to plug in for the applications of 1990s and beyond. Clients and servers typically communicate with each other by using one of the following paradigms:

- Remote Procedure Call (RPC). In this paradigm, the client process invokes a remotely located procedure (a server process), the remote procedure executes

and sends the response back to the client. The remote procedure can be simple (e.g., retrieve time of day) or complex (e.g., retrieve all customers from Chicago who have a good credit rating). Each request/response of an RPC is treated as a separate unit of work, thus each request must carry enough information needed by the server process. RPCs are supported widely at present.

- Remote Data Access (RDA). This paradigm allows client programs and/or end-user tools to issue ad hoc queries, usually SQL, against remotely located databases. The key technical difference between RDA and RPC is that in an RDA the size of the result is not known because the result of an SQL query could be one row or thousands of rows. RDA is heavily supported by database vendors.

- Queued Message Processing (QMP). In this paradigm, the client message is stored in a queue and the server works on it when free. The server stores ("puts") the response in another queue and the client actively retrieves ("gets") the responses from this queue. This model, used in many transaction processing systems, allows the clients to asynchronously send requests to the server. Once a request is queued, the request is processed even if the sender is disconnected (intentionally or due to a failure). QMP support is becoming commonly available.

Initial implementations of client/server architecture were based on the "two-tiered" architectures shown in Figure 2.5 (a) through Figure 2.5 (e) (these architectural configurations are known as the "Gartner Group" configurations). The first two architectures (Figure 2.5 (a) and Figure 2.5 (b) are used in many presentation intensive applications (e.g., XWindow, multimedia presentations) and to provide a "face lift" to legacy applications by building a GUI interface that invokes the older text-based user interfaces of legacy applications. Figure 2.5 (c) represents the distributed application program architecture in which the application programs are split between the client and server machines, and they communicate with each other through the remote procedure

call (RPC) or queued messaging middleware. Figure 2.5 (d) represents the remote data architecture in which the remote data is typically stored in a "SQL server" and is accessed through ad hoc SQL statements sent over the network. Figure 2.5 (e) represents the case where the data exist at client as well as server machines (distributed data architecture).

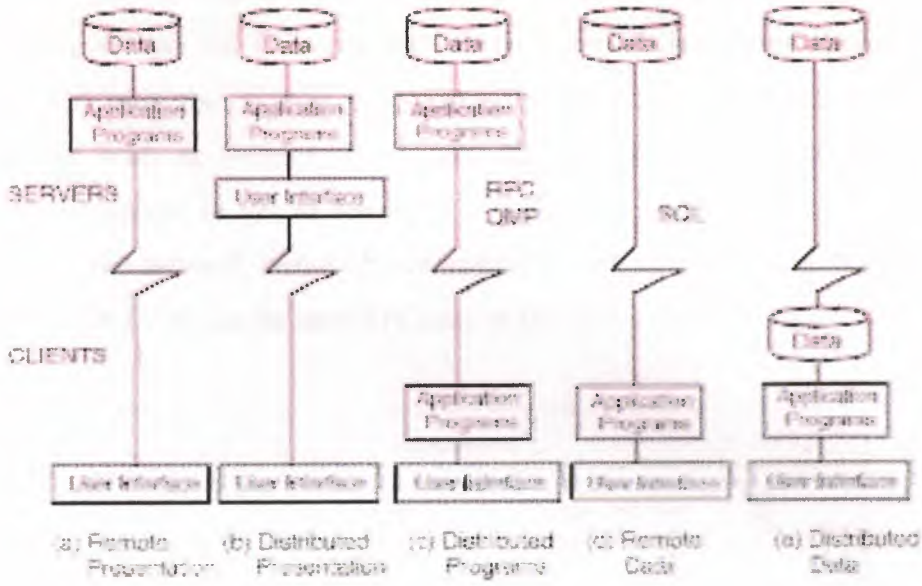


Figure 2.5 Traditional Client/Server Architectures

Although a given C/S application can be architected in any of these configurations, the remote data and distributed program configurations are used heavily at present. The remote data configuration at present is very popular for departmental applications and is heavily supported by numerous database vendors (as a matter of fact this configuration is used to represent typical two-tiered architectures that rely on remote SQL). Most data warehouses also use a remote data configuration because the data warehouse tools can reside on user workstations and issue remote SQL calls to the data warehouse. However, the distributed programs configuration is very useful for enterprisewide applications, because the application programs on both sides can exchange information through messages.

2.5.1 OSF DCE–A Client/Server Environment

The Open Software Foundation (OSF) Distributed Computing Environment (DCE) packages and implements "open" and de facto standards into an environment for distributed client/server computing. OSF DCE, also commonly known as DCE, is currently available on a wide range of computing platforms such as UNIX, OS/2, and IBM MVS. Figure 2.6 shows a conceptual view of OSF DCE. The applications are at the highest level and the OSI transport services are at the lowest level in DCE (at present, DCE uses the TCP/IP transport services). The security and management functions are built at various levels and are applicable to all components. The distributed file access to get at remotely located data, naming services for accessing objects across the network, remote procedure calls (RPCs), and presentation services are at the core of DCE. As can be seen RPCs are at the core of DCE.

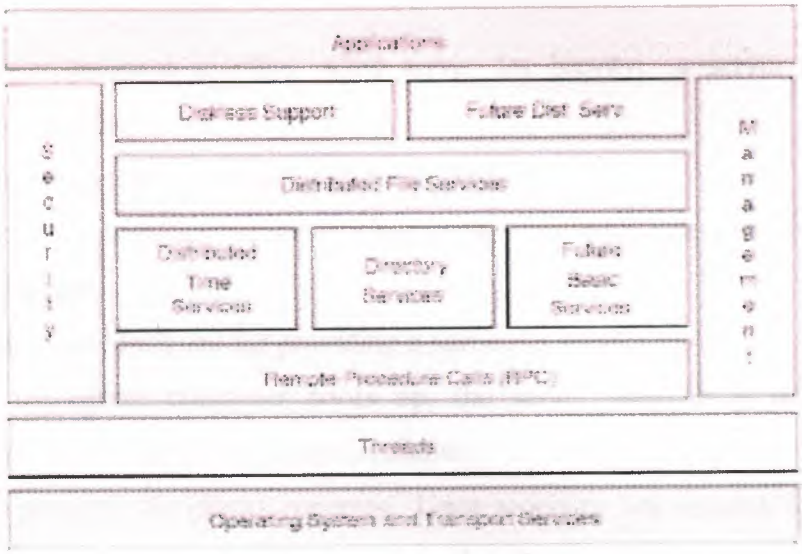


Figure 2.6 OSF DCE

2.6 The Relationship Between Client And Server

Is the fundamental paradigm that determines the characteristics of computer languages on the web. The differences are played out in terms of:

- *How the languages are invoked*
- *What resources and information they can access*
- *What structures can be manipulated*

- *How they interact with the user and system*
- *The speed and predictability of their interactions*

2.7 Client Server Model

1- Transport protocols (such as TCP/IP) only provide mechanisms to transfer data.

2- Application programs use these mechanisms to provide various functionalities, such as electronic mail, file transfer, and network games.

3- Many application programs use the client-server paradigm for interaction through the networks.

2.7.1 Client-Server Paradigm

1- An application is composed of a *server program* and a *client program.* These two programs communicate through networks.

2- Remark

* The server program is often called *server* for brevity; a server is a piece of software, not some hardware.

* The client program is also called *client* for brevity.

* The computer on which the server program is executed is called *server computer.*

3- Server Program

* It is a program designed for providing a particular service.

* When the server computer boots up, the server program is usually invoked automatically, i.e., the service is always available.

* It waits passively for clients' requests. Upon receiving one request, it performs the necessary computation and returns the result to the client.

* In many applications, a server is designed such that it can serve multiple clients simultaneously (e.g., web servers).

  - It usually requires powerful hardware.

  - It requires an operating system that supports multitasking.

4- **Client Program**

* It is an application program executed by an end user on a local computer.

* When it needs a service, it sends a request to the server program, and then waits for the response from the server program.

* It can request multiple services when it is necessary.

2.7.2 Client Server Interaction

1- The client and the server use a transport protocol to send and receive data. For example, they can use the TCP/IP protocol suite to communicate through the internet:

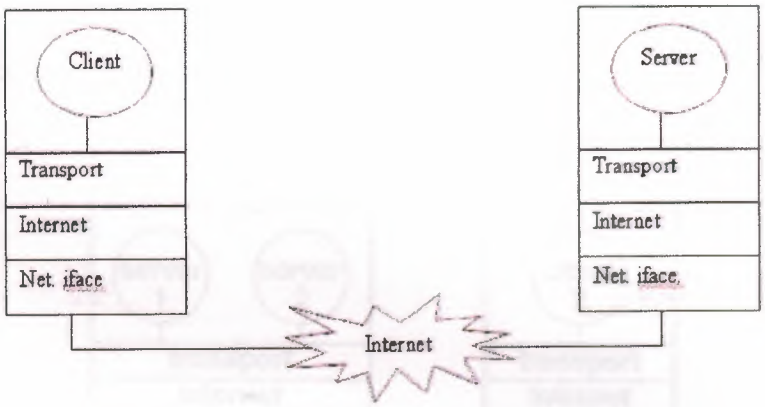


Figure 2.8 Client Server Interaction

2- Two classes of interactions between the server and the client:

* **Connection-oriented** interaction using TCP
* **Connectionless** interaction using UDP

3- Remark

* TCP provides reliable data delivery across the internet, while UDP does not guarantee reliable data delivery.

- Using connection-oriented interaction, the network programmer need not worry about data reliability.
- Programming is simpler.

4- The client and the server can interact in many possible manners, e.g.,

* An application (a client) interacts with one server only.
* An application becomes a client of one service, and later becomes a client of another service.

- For example, an application requests print service to print a file, and then requests file service to save the file.

*A server for one service can become a client of another.

-For example, a file server that needs to record the time of file access can become a client of a time server.

2.7.3 Multiple Services

* A powerful computer can run multiple server programs to provide multiple services at the same time.

* Example: a computer runs a ftp server and a web server.

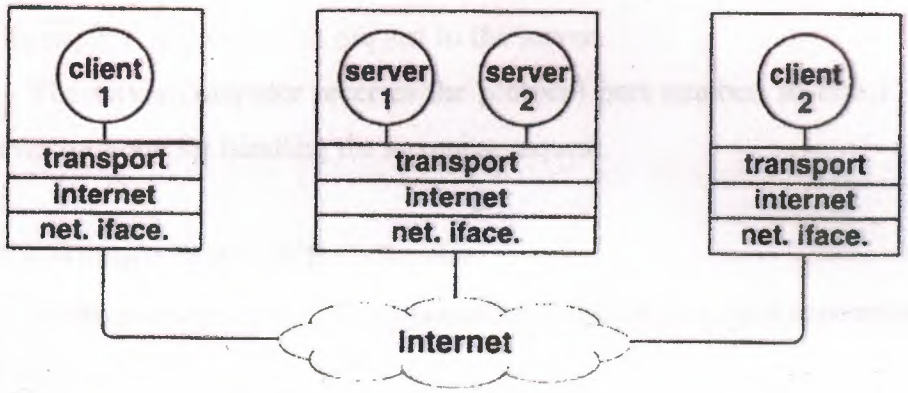


Figure 2.9 Multiple Services

* To provide multiple services,

- the computer must have sufficient resources (e.g., fast processor(s) and large memory);
- the OS supports multitasking (e.g., UNIX, Windows).

* Service Identification Problem

- Suppose a computer is running multiple server programs.
- When this computer receives a client's request, how can it identify the requested service?

* Service Identification Solution

- When a server program begins execution, it registers with the local protocol software and specifies a unique identifier for its service.

- When a client requests a particular service, it sends the identifier of this service to the server.

- The server computer receives the identifier, so that it can identify the server program for handling this request.

* Service Identification in TCP

- In TCP, each service is identified by a unique **16-bit protocol port number**.

- When a client requests a service, it specifies the protocol port number of this service in its request, and sends this request to the server.

- The server computer receives the protocol port number, so that it can identify the server program for handling the incoming request.

2.7.4 Multiple Protocols for a Service

* A service can use either a connection-oriented transport, or a connectionless transport, or both.

* When a service supports two or more transport protocols, the client can choose the transport protocol.

* Two possible implementations:

- Implement one server program for connection-oriented transport, and implement another for connectionless transport.

- Implement one server program that can provide different transport for different clients' requests.

2.8 Client/Server Computing

To truly understand how much of the Internet operates, including the Web, it is important to understand the concept of client/server computing. The client/server model is a form of distributed computing where one program (the client) communicates with another program (the server) for the purpose of exchanging information.

The client's responsibility is usually to:

1. Handle the user interface.
2. Translate the user's request into the desired protocol.
3. Send the request to the server.
4. Wait for the server's response.
5. Translate the response into "human-readable" results.
6. Present the results to the user.

The server's functions include:

1. Listen for a client's query.
2. Process that query.
3. Return the results back to the client.

A typical client/server interaction goes like this:

1. The user runs client software to create a query.
2. The client connects to the server.
3. The client sends the query to the server.
4. The server analyzes the query.
5. The server computes the results of the query.
6. The server sends the results to the client.
7. The client presents the results to the user.
8. Repeat as necessary.

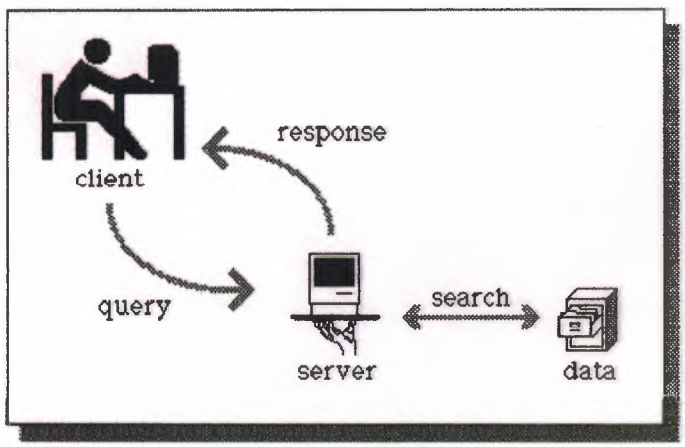


Figure 2.10 A typical client/server interaction Computing

This client/server interaction is a lot like going to a French restaurant. At the restaurant, you (the user) are presented with a menu of choices by the waiter (the client). After making your selections, the waiter takes note of your choices, translates them into French, and presents them to the French chef (the server) in the kitchen. After the chef prepares your meal, the waiter returns with your diner (the results). Hopefully, the waiter returns with the items you selected, but not always; sometimes things get "lost in the translation."

Flexible user interface development is the most obvious advantage of client/server computing. It is possible to create an interface that is independent of the server hosting the data. Therefore, the user interface of a client/server application can be written on a Macintosh and the server can be written on a mainframe. Clients could be also written for DOS- or UNIX-based computers. This allows information to be stored in a central server and disseminated to different types of remote computers. Since the user interface is the responsibility of the client, the server has more computing resources to spend on analyzing queries and disseminating information. This is another major advantage of client/server computing; it tends to use the strengths of divergent computing platforms to create more powerful applications. Although its computing and storage capabilities are dwarfed by those of the mainframe, there is no reason why a Macintosh could not be used as a server for less demanding applications.

In short, client/server computing provides a mechanism for disparate computers to cooperate on a single computing task.

Note: *In Chapter III I will cover all the protocols of Client-Server. It will be explained more than this, And i will also cover all the information about TCP / IP.*

CHAPTER THREE
TCP/IP AND THE INTERNET

3.1 Introduction

Before proceeding into a considerable amount of detail about TCP/IP, the Internet, and the Internet Protocol (IP), it is worthwhile to try to complete a quick outline of TCP/IP. Then, as the details of each protocol are discussed individually, they can be placed in the broader outline more easily, just what is TCP/IP? Protocol used a software-based communications in networking. Although the name TCP/IP implies that the entire scope of the product is a combination of two protocols Transmission Control Protocol and Internet Protocol, the term TCP/IP refers not to a single entity combining two protocols, but a larger set of software programs that provides network services such as remote logins, remote file transfers, and electronic mail. TCP/IP provides a method for transferring information from one machine to another. A communications protocol should handle errors in transmission, manage the routing and delivery of data, and control the actual transmission by the use of predetermined status signals. TCP/IP accomplishes all of this.

TCP/IP is not a single product. It is a catchall name for a family of protocols that use a similar behavior. Using the term TCP/IP usually refers to one or more protocols within the family, not just TCP and IP.

The OSI Reference Model is composed of seven layers. TCP/IP was designed with layers as well, although they do not correspond one-to-one with the OSI-RM layers. You can overlay the TCP/IP programs on this model to give you an idea of where all the TCP/IP layers reside. I do that in a little more detail later in this chapter. Before that, I take a quick look at the TCP/IP protocols and how they relate to each other, and show a rough mapping to the OSI layers. Figure 3.1 shows the basic elements of the TCP/IP family of protocols. You can see that TCP/IP is not involved in the bottom two layers of the OSI model (data link and physical) but begins in the network layer, where the Internet Protocol (IP) resides. In the transport layer, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are involved. Above this, the utilities and protocols that make up the rest of the TCP/IP suite are built using the TCP or UDP and IP layers for their communications system.

Telnet - RemoteLogin
FTP - File Transfer Protocol
SMTP - Simple Mail Transfer Protocol
X - X Windows System
Kerberos - Security
DNS - Domain Name System
ASN - Abstract Syntax Notation
SNMP - Simple Network Management Protocol

NFS - Network File Server
RPC - Remote Procedure Calls
TFTP - Trivial File Transfer Protocol
TCP - Transmission Control Protocol
User Datagram Protocol
IP - Internet Protocol
ICMP - Internet Control Message Protocol

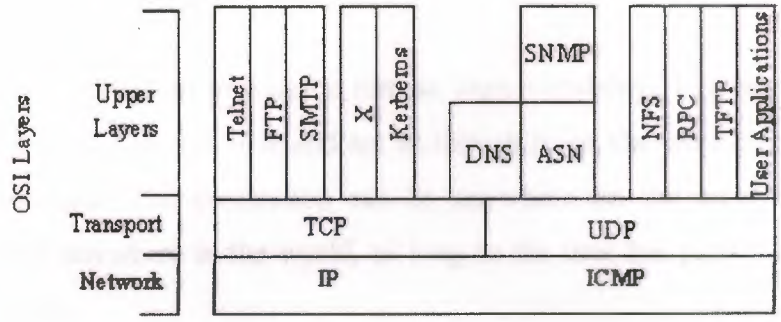


Figure 3.1 TCP/IP Suite and OSI layers.

Figure 3.1 shows that some of the upper-layer protocols depend on TCP (such as Telnet and FTP), whereas some depend on UDP (such as TFTP and RPC). Most upper-layer TCP/IP protocols use only one of the two transport protocols (TCP or UDP), although a few, including DNS (Domain Name System) can use both. A note of caution about TCP/IP: Despite the fact that TCP/IP is an open protocol, many companies have modified it for their own networking system. There can be incompatibilities because of these modifications, which, even though they might adhere to the official standards, might have other aspects that cause problems. Luckily, these types of changes are not rampant, but you should be careful when choosing a TCP/IP product to ensure its compatibility with existing software and hardware. TCP/IP is dependent on the concept of clients and servers. This has nothing to do with a file server being accessed by a diskless workstation or PC. The term client/server has a simple meaning in TCP/IP: any device that initiates communications is the client, and the device that answers is the server. The server is responding to (serving) the client's requests.

3.2 A quick Overview of TCP/IP Components

To understand the roles of the many components of the TCP/IP protocol family, it is useful to know what you can do over a TCP/IP network. Then, once the applications are understood, the protocols that make it possible are a little easier to comprehend. The following list is not exhaustive but mentions the primary user applications, which the TCP/IP network provides.

3.2.1 Telnet

The Telnet program provides a remote login capability. This lets a user on one machine log onto another machine and act as though he or she were directly in front of the second machine. The connection can be anywhere on the local network or on another network anywhere in the world, as long as the user has permission to log onto the remote system.

You can use Telnet when you need to perform actions on a machine across the country. This isn't often done except in a LAN or WAN context, but a few systems accessible through the Internet allow Telnet sessions while users play around with a new application or operating system.

3.2.2 File Transfer Protocol

File Transfer Protocol (FTP) enables a file on one system to be copied to another *system. The user does not actually log in as a full user to the machine he or she wants to* access, as with Telnet, but instead uses the FTP program to enable access. Again, the correct permissions are necessary to provide access to the files. Once the connection to a remote machine has been established, FTP enables you to copy one or more files to your machine. (The term *transfer* implies that the file is moved from one system to another but the original is not affected. Files are copied.) FTP is a widely used service on the Internet, as well as on many large LANs and WANs.

3.2.3 Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP) is used for transferring electronic mail. SMTP is completely transparent to the user. Behind the scenes, SMTP connects to remote machines and transfers mail messages much like FTP transfers files. Users are

almost never aware of SMTP working, and few system administrators have to bother with it. SMTP is a mostly trouble-free protocol and is in very wide use.

3.3.4 Kerberos

Kerberos is a widely supported security protocol. Kerberos uses a special application called an *authentication server* to validate passwords and encryption schemes. Kerberos is one of the more secure encryption systems used in communications and is quite common in UNIX.

3.2.5 Domain Name System

Domain Name System (DNS) enables a computer with a common name to be converted to a special network address. For example, another machine on the same network (or any other connected network) cannot access a PC called Darkstar unless some method of checking the local machine name and replacing the name with the machine's hardware address is available. DNS provides a conversion from the common local name to the unique physical address of the device's network connection.

3.2.6 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) provides status messages and problem reports across a network to an administrator. SNMP uses User Datagram Protocol (UDP) as a transport mechanism. SNMP employs slightly different terms from TCP/IP, working with managers and agents instead of clients and servers (although they mean essentially the same thing). An agent provides information about a device, whereas a manager communicates across a network with agents.

3.2.7 Network File System

Network File System (NFS) is a set of protocols developed by Sun Microsystems to enable multiple machines to access each other's directories transparently. They accomplish this by using a distributed file system scheme. NFS systems are common in large corporate environments, especially those that use UNIX workstations.

3.2.8 Remote Procedure Call

The Remote Procedure Call (RPC) protocol is a set of functions that enable an application to communicate with another machine (the server). It provides for programming functions, return codes, and predefined variables to support distributed computing.

3.2.9 Trivial File Transfer Protocol

Trivial File Transfer Protocol (TFTP) is a very simple, unsophisticated file transfer protocol that lacks security. It uses UDP as a transport. TFTP performs the same task as FTP, but uses a different transport protocol.

3.2.10 Transmission Control Protocol

Transmission Control Protocol (the TCP part of TCP/IP) is a communications protocol that provides reliable transfer of data. It is responsible for assembling data passed from higher-layer applications into standard packets and ensuring that the data is transferred correctly.

3.2.11 User Datagram Protocol

User Datagram Protocol (UDP) is a connectionless-oriented protocol, meaning that it does not provide for the retransmission of datagram's (unlike TCP, which is connection-oriented). UDP is not very reliable, but it does have specialized purposes. If the applications that use UDP have reliability checking built into them, the shortcomings of UDP are overcome.

3.2.12 Internet Protocol

Internet Protocol (IP) is responsible for moving the packets of data assembled by either TCP or UDP across networks. It uses a set of unique addresses for every device on the network to determine routing and destinations.

3.2.13 Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is responsible for checking and generating messages on the status of devices on a network. It can be used to inform other devices of a failure in one particular machine. ICMP and IP usually work together.

3.3 TCP/IP History

The architecture of TCP/IP is often called the Internet architecture because TCP/IP and the Internet as so closely interwoven. In the last chapter, you saw how the Internet standards were developed by the Defense Advanced Research Projects Agency (DARPA) and eventually passed on to the Internet Society.

The Internet was originally proposed by the precursor of DARPA, called the Advanced Research Projects Agency (ARPA), as a method of testing the viability of packet-switching networks. (When ARPA's focus became military in nature, the name was changed.) During its tenure with the project, ARPA foresaw a network of leased lines connected by switching nodes. The network was called ARPANET, and the switching nodes were called Internet Message Processors, or IMPs. The ARPANET was initially to be comprised of four IMPs located at the University of California at Los Angeles, the University of California at Santa Barbara, the Stanford Research Institute, and the University of Utah. The original IMPs were to be Honeywell 316 minicomputers. Bolt, Beranek, and Newman (BBN), a company that had a strong influence on the development of the network in the following years, won the contract for the installation of the network. The contract was awarded in late 1968, followed by testing and refinement over the next five years.

In 1971, ARPANET entered into regular service. Machines used the ARPANET by connecting to an IMP using the "1822" protocol so called because that was the number of the technical paper describing the system. During the early years, the purpose and utility of the network was widely (and sometimes heatedly) discussed, leading to refinements and modifications as users requested more functionality from the system.

A commonly recognized need was the capability to transfer files from one machine to another, as well as the capability to support remote logins. Remote logins would enable a user in Santa Barbara to connect to a machine in Los Angeles over the network and function as though he or she were in front of the UCLA machine. The protocol then in use on the network was not capable of handling these new functionality requests; so new protocols were continually developed, refined, and tested. Remote login and remote file transfer were finally implemented in a protocol called the Network Control Program (NCP). Later, electronic mail was added through File Transfer Protocol (FTP). Together with NCP's remote logins and file transfer, this formed the basic services for ARPANET.

By 1973, it was clear that NCP was unable to handle the volume of traffic and proposed new functionality. A project was begun to develop a new protocol. The TCP/IP and gateway architectures were first proposed in 1974. The published article by Cerf and Kahn described a system that provided a standardized application protocol that also used end-to-end acknowledgments.

Neither of these concepts was really novel at the time, but more importantly (and with considerable vision), Cerf and Kahn suggested that the new protocol be independent of the underlying network and computer hardware. Also, they proposed universal connectivity throughout the network. These two ideas were radical in a world of proprietary hardware and software, because they would enable any kind of platform to participate in the network. The protocol was developed and became known as TCP/IP.

A series of RFCs (Requests for Comment, part of the process for adopting new Internet Standards) was issued in 1981, standardizing TCP/IP version 4 for the ARPANET. In 1982, TCP/IP supplanted NCP as the dominant protocol of the growing network, which was now connecting machines across the continent. It is estimated that a new computer was connected to ARPANET every 20 days during its first decade. (That might not seem like much compared to the current estimate of the Internet's size doubling every year, but in the early 1980s it was a phenomenal growth rate.)

During the development of ARPANET, it became obvious that nonmilitary researchers could use the network to their advantage, enabling faster communication of ideas as well as faster physical data transfer. A proposal to the National Science Foundation lead to funding for the Computer Science Network in 1981, joining the military with educational and research institutes to refine the network. This led to the splitting of the network into two different networks in 1984. MILNET was dedicated to unclassified military traffic, whereas ARPANET was left for research and other nonmilitary purposes. ARPANET's growth and subsequent demise came with the approval for the Office of Advanced Scientific Computing to develop wide access to supercomputers. They created NSFNET to connect six supercomputers spread across the country through T-1 lines (which operated at 1.544 Mbps). The Department of Defense finally declared ARPANET obsolete in 1990, when it was officially dismantled.

3.4 Berkeley UNIX Implementations and TCP/IP

TCP/IP became important when the Department of Defense started including the protocols as military standards, which were required for many contracts. TCP/IP became popular primarily because of the work done at UCB (Berkeley). UCB had been a center of UNIX development for years, but in 1983 they released a new version that incorporated TCP/IP as an integral element. That version 4.2BSD (Berkeley System Distribution) was made available to the world as public domain software. The popularity of 4.2BSD spurred the popularity of TCP/IP, especially as more sites connected to the growing ARPANET. Berkeley released an enhanced version (which included the so-called Berkeley Utilities) in 1986 as 4.3BSD. An optimized TCP implementation followed in 1988 (4.3BSD/Tahoe). Practically every version of TCP/IP available today has its roots (and much of its code) in the Berkeley versions. Despite the demise of Berkeley Software Distribution's UNIX version in 1993, the BSD and UCB developments are integral parts of TCP/IP and continue to be used as part of the protocol family's naming system.

3.5 OSI and TCP/IP

The adoption of TCP/IP didn't conflict with the OSI standards because the two developed concurrently. In some ways, TCP/IP contributed to OSI, and vice-versa. Several important differences do exist, though, which arise from the basic requirements of TCP/IP, which are:

- A common set of applications
- Dynamic routing
- Connectionless protocols at the networking level
- Universal connectivity
- Packet-switching

The differences between the OSI architecture and that of TCP/IP relate to the layers above the transport level and those at the network level. OSI has both the session layer and the presentation layer, whereas TCP/IP combines both into an application layer. The requirement for a connectionless protocol also required TCP/IP to combine OSI's physical layer and data link layer into a network level. TCP/IP also includes the session

and presentation layers of the OSI model into TCP/IP's application layer. A schematic view of TCP/IP's layered structure compared with OSI's seven-layer model is shown in Figure 3.2. TCP/IP calls the different network level elements subnetworks.

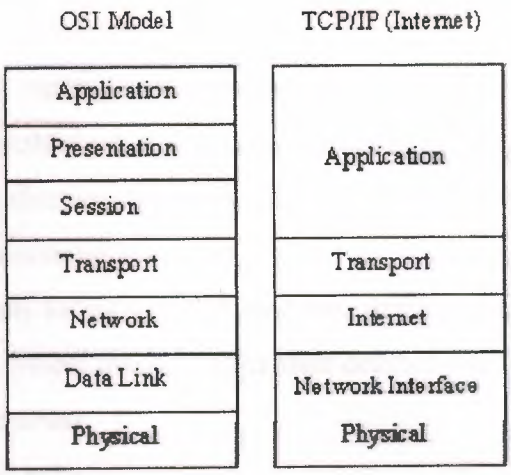


Figure 3.2. The OSI and TCP/IP Layered Structures.

OSI and TCP/IP are not incompatible, but neither are they perfectly compatible. They both have a layered architecture, but the OSI architecture is much more rigorously defined, and the layers are more independent than TCP/IP's.

Some fuss was made about the network level combination, although it soon became obvious that the argument was academic, as most implementations of the OSI model combined the physical and link levels on an intelligent controller (such as a network card). The combination of the two layers into a single layer had one major benefit: it enabled a subnetwork to be designed that was independent of any network protocols, because TCP/IP was oblivious to the details. This enabled proprietary, self-contained networks to implement the TCP/IP protocols for connectivity outside their closed systems.

The layered approach gave rise to the name TCP/IP. The transport layer uses the Transmission Control Protocol (TCP) or one of several variants, such as the User Datagram Protocol (UDP). (There are other protocols in use, but TCP and UDP are the most common.) There is, however, only one protocol for the network level: the Internet

Protocol (IP). This is what assures the system of universal connectivity, one of the primary design goals.

There is a considerable amount of pressure from the user community to abandon the OSI model (and any future communications protocols developed that conform to it) in favor of TCP/IP. The argument hinges on some obvious reasons:

- TCP/IP is up and running and has a proven record.
- TCP/IP has an established, functioning management body.
- Thousands of applications currently use TCP/IP and its well-documented application programming interfaces.
- TCP/IP is the basis for most UNIX systems, which are gaining the largest share of the operating system market (other than desktop single-user machines such as the PC and Macintosh).
- TCP/IP is vendor-independent.

Arguing rather strenuously against TCP/IP, surprisingly enough, is the US government the very body that sponsored it in the first place. Their primary argument is that TCP/IP is not an internationally adopted standard, whereas OSI has that recognition. The Department of Defense has even begun to move its systems away from the TCP/IP protocol set. A compromise will probably result, with some aspects of OSI adopted into the still-evolving TCP/IP protocol suite.

3.6 TCP/IP and Ethernet

For many people the terms TCP/IP and Ethernet go together almost automatically, primarily for historical reasons, as well as the simple fact that there are more Ethernet-based TCP/IP networks than any other type. Ethernet was originally developed at Xerox's Palo Alto Research Center as a step toward an electronic office communications system, and it has since grown in capability and popularity.

Ethernet is a hardware system providing for the data link and physical layers of the OSI model. As part of the Ethernet standards, issues such as cable type and broadcast speeds are established. There are several different versions of Ethernet, each with a different data transfer rate. The most common is Ethernet version 2, also called 10Base5, Thick Ethernet, and IEEE 802.3 (after the number of the standard that defines the system

adopted by the Institute of Electrical and Electronic Engineers). This system has a 10 Mbps rate.

There are several commonly used variants of Ethernet, such as Thin Ethernet (called 10Base2), which can operate over thinner cable (such as the coaxial cable used in cable television systems), and Twisted-Pair Ethernet (10BaseT), which uses simple twisted-pair wires similar to telephone cable. The latter variant is popular for small companies because it is inexpensive, easy to wire, and has no strict requirements for distance between machines.

It is usually easy to tell which type of Ethernet network is being used by checking the connector to a network card. If it has a telephone-style plug, it is 10BaseT. The cable for 10BaseT looks the same as telephone cable. If the network has a D-shaped connector with many pins in it, it is 10Base5. A 10Base2 network has a connector similar to a cable TV coaxial connector, except it locks into place. The10Base2 connector is always circular. The size of a network is also a good indicator. 10Base5 is used in large networks with many devices and long transmission runs. 10Base2 is used in smaller networks, usually with all the network devices in fairly close proximity. Twisted-pair (10BaseT) networks are often used for very small networks with a maximum of a few dozen devices in close proximity.

Ethernet and TCP/IP work well together, with Ethernet providing the physical cabling (layers one and two) and TCP/IP the communications protocol (layers three and four) that is broadcast over the cable. The two have their own processes for packaging information: TCP/IP uses 32-bit addresses, whereas Ethernet uses a 48-bit scheme. The two work together, however, because of one component of TCP/IP called the Address Resolution Protocol (ARP), which converts between the two schemes. (I discuss ARP in more detail later, in the section titled "Address Resolution Protocol.")

Ethernet relies on a protocol called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). To simplify the process, a device checks the network cable to see if anything is currently being sent. If it is clear, the device sends its data. If the cable is busy (carrier detect), the device waits for it to clear. If two devices transmit at the same time (a collision), the devices know because of their constant comparison of the cable traffic to the data in the sending buffer. If a collision occurs, the devices wait a random amount of time before trying again.

3.7 The Internet

As ARPANET grew out of a military-only network to add subnetworks in universities, corporations, and user communities, it became known as the Internet. There is no single network called the Internet, however. The term refers to the collective network of subnetworks. The one thing they all have in common is TCP/IP as a communications protocol. The organization of the Internet and adoption of new standards is controlled by the Internet Advisory Board (IAB). Among other things, the IAB coordinates several task forces, including the Internet Engineering Task Force (IETF) and Internet Research Task Force (IRTF). In a nutshell, the IRTF is concerned with ongoing research, whereas the IETF handles the implementation and engineering aspects associated with the Internet.

A body that has some bearing on the IAB is the Federal Networking Council (FNC), which serves as an intermediary between the IAB and the government. The FNC has an advisory capacity to the IAB and its task forces, as well as the responsibility for managing the government's use of the Internet and other networks. Because the government was responsible for funding the development of the Internet, it retains a considerable amount of control, as well as sponsoring some research and expansion of the Internet.

3.7.1 The Structure of the Internet

As mentioned earlier, the Internet is not a single network but a collection of networks that communicate with each other through gateways. For the purposes of this chapter, a *gateway* (sometimes called a *router*) is defined as a system that performs relay functions between networks, as shown in Figure 3.3. The different networks connected to each other through gateways are often called subnetworks, because they are a smaller part of the larger overall network. This does not imply that a subnetwork is small or dependent on the larger network. Subnetworks are complete networks, but they are connected through a gateway as a part of a larger internetwork, or in this case the Internet.

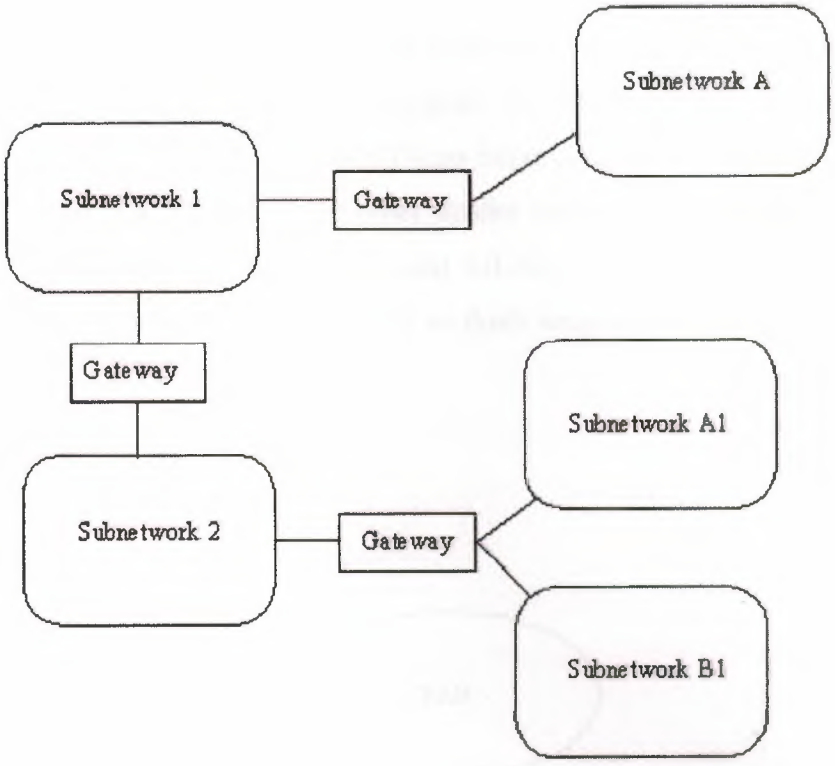


Figure 3.3. Gateways Act As Relays Between Subnetworks.

With TCP/IP, all interconnections between physical networks are through gateways. An important point to remember for use later is that gateways route information packets based on their destination network name, not the destination machine. Gateways are supposed to be completely transparent to the user, which alleviates the gateway from handling user applications (unless the machine that is acting as a gateway is also someone's work machine or a local network server, as is often the case with small networks). Put simply, the gateway's sole task is to receive a Protocol Data Unit (PDU) from either the internetwork or the local network and either route it on to the next gateway or pass it into the local network for routing to the proper user.

Gateways work with any kind of hardware and operating system, as long as they are designed to communicate with the other gateways they are attached to (which in this case means that it uses TCP/IP). Whether the gateway is leading to a Macintosh network, a set of IBM PCs, or mainframes from a dozen different companies doesn't matter to the gateway or the PDUs it handles.

There are actually several types of gateways, each performing a difference type of task.

In the United States, the Internet has the NFSNET as its backbone, as shown in Figure 3.4. Among the primary networks connected to the NFSNET are NASA's Space Physics Analysis Network (SPAN), the Computer Science Network (CSNET), and several other networks such as WESTNET and the San Diego Supercomputer Network (SDSCNET), not shown in Figure 3.4. There are also other smaller user-oriented networks such as the Because It's Time Network (BITNET) and UUNET, which provide connectivity through gateways for smaller sites that can't or don't want to establish a direct gateway to the Internet.

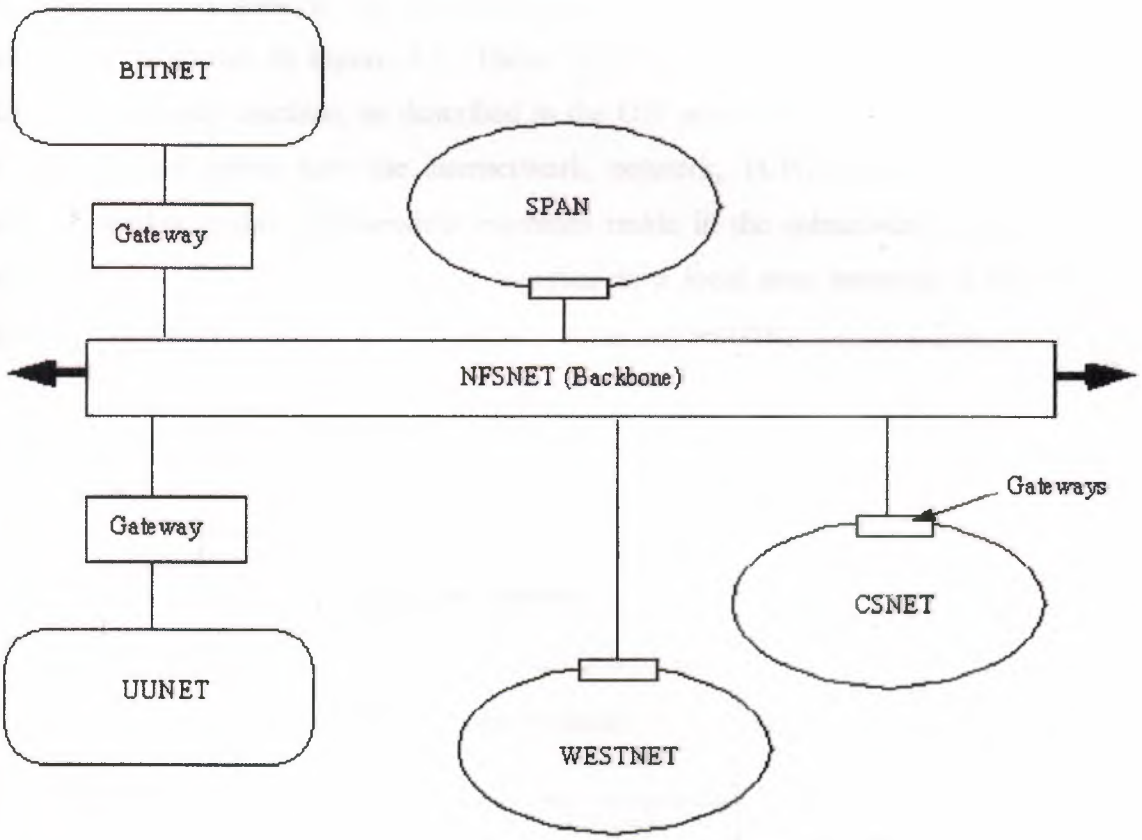


Figure 3.4. The US Internet Network.

The NFSNET backbone is comprised of approximately 3,000 research sites, connected by T-3 leased lines running at 44.736 Megabits per second. Tests are currently underway to increase the operational speed of the backbone to enable more throughput and accommodate the rapidly increasing number of users. Several technologies are being field-tested, including Synchronous Optical Network (SONET), Asynchronous Transfer Mode (ATM), and ANSI's proposed High-Performance Parallel Interface

(HPPI). These new systems can produce speeds approaching 1 Gigabit per second.

3.7.2 The Internet Layers

Most internetworks, including the Internet, can be thought of as a layered architecture (yes, even more layers!) to simplify understanding. The layer concept helps in the task of developing applications for internetworks. The layering also shows how the different parts of TCP/IP work together, so applying it to the Internet makes sense. Be careful to think of these layers as conceptual only; they are not really physical or software layers as such (unlike the OSI or TCP/IP layers).

It is convenient to think of the Internet as having four layers. This layered Internet architecture is shown in Figure 3.5. These layers should not be confused with the architecture of each machine, as described in the OSI seven-layer model. Instead, they are a method of seeing how the internetwork, network, TCP/IP, and the individual machines work together. Independent machines reside in the subnetwork layer at the bottom of the architecture, connected together in a local area network (LAN) and referred to as the subnetwork, a term you saw in the last section.

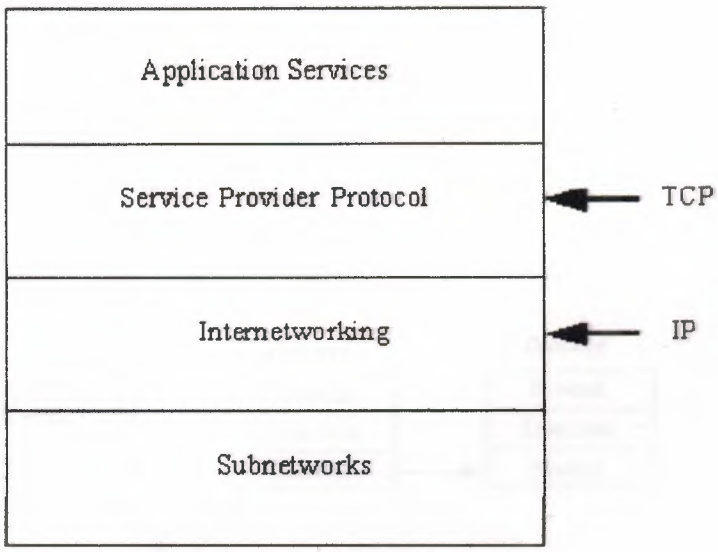


Figure 3.5. The Internet Architecture.

On top of the subnetwork layer is the internetwork layer, which provides the functionality for communications between networks through gateways. Each subnetwork uses gateways to connect to the other subnetworks in the internetwork. The internetwork layer is where data gets transferred from gateway to gateway until it reaches its destination and then passes into the subnetwork layer. The internetwork layer runs the Internet Protocol (IP).

The service provider protocol layer is responsible for the overall end-to-end communications of the network. This is the layer that runs the Transmission Control Protocol (TCP) and other protocols. It handles the data traffic flow itself and ensures reliability for the message transfer. The top layer is the application services layer, which supports the interfaces to the user applications. This layer interfaces to electronic mail, remote file transfers, and remote access. Several protocols are used in this layer, many of which you will read about later. To see how the Internet architecture model works, a simple example is useful. Assume that an application on one machine wants to transfer a datagram to an application on another machine in a different subnetwork. Without all the signals between layers, and simplifying the architecture a little, the process is shown in Figure 3.6. The layers in the sending and receiving machines are the OSI layers, with the equivalent Internet architecture layers indicated.

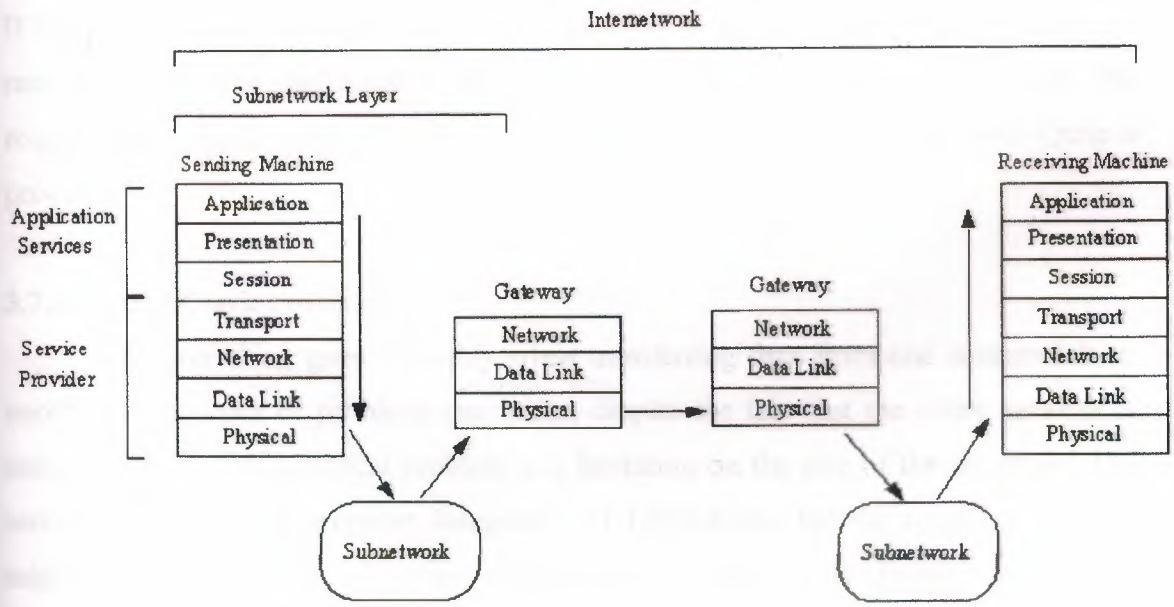


Figure 3.6. Transfer Of A Datagram Over An Internetwork.

The data is sent down the layers of the sending machine, assembling the datagram with the Protocol Control Information (PCI) as it goes. From the physical layer, the datagram (which is sometimes called a *frame* after the data link layer has added its header and trailing information) is sent out to the local area network. The LAN routes the information to the gateway out to the internetwork. During this process, the LAN has no concern about the message contained in the datagram. Some networks, however, alter the header information to show, among other things, the machines it has passed through. From the gateway, the frame passes from gateway to gateway along the internetwork until it arrives at the destination subnetwork. At each step, the gateway analyzes the datagram's header to determine if it is for the subnetwork the gateway leads to. If not, it routes the datagram back out over the internetwork. This analysis is performed in the physical layer, eliminating the need to pass the frame up and down through different layers on each gateway. The header can be altered at each gateway to reflect its routing path. When the datagram is finally received at the destination subnetwork's gateway, the gateway recognizes that the datagram is at its correct subnetwork and routes it into the LAN and eventually to the target machine. The routing is accomplished by reading the header information. When the datagram reaches the destination machine, it passes up through the layers, with each layer stripping off its PCI header and then passing the result on up. At long last, the application layer on the destination machine processes the final header and passes the message to the correct application.

If the datagram was not data to be processed but a request for a service, such as a remote file transfer, the correct layer on the destination machine would decode the request and route the file back over the internetwork to the original machine. Quite a process!

3.7.3 Internetwork Problems

Not everything goes smoothly when transferring data from one subnetwork to another. All manner of problems can occur, despite the fact that the entire network is using one protocol. A typical problem is a limitation on the size of the datagram. The sending network might support datagram's of 1,024 bytes, but the receiving network might use only 512-byte datagram's (because of a different hardware protocol, for example). This is where the processes of segmentation, separation, reassembly, and concatenation (explained in the last chapter) become important.

The actual addressing methods used by the different subnetworks can cause conflicts when routing datagram's. Because communicating subnetworks might not have the same network control software, the network-based header information might differ, despite the fact that the communications methods are based on TCP/IP. An associated problem occurs when dealing with the differences between physical and logical machine names. In the same manner, a network that requires encryption instead of clear-text datagram's can affect the decoding of header information. Therefore, differences in the security implemented on the subnetworks can affect datagram traffic. These differences can all be resolved with software, but the problems associated with addressing methods can become considerable.

Another common problem is the different networks' tolerance for timing problems. Time-out and retry values might differ, so when two subnetworks are trying to establish communication, one might have given up and moved on to another task while the second is still waiting patiently for an acknowledgment signal. Also, if two subnetworks are communicating properly and one gets busy and has to pause the communications process for a short while, the amount of time before the other network assumes a disconnection and gives up might be important. Coordinating the timing over the internetwork can become very complicated.

Routing methods and the speed of the machines on the network can also affect the internetwork's performance. If a gateway is managed by a particularly slow machine, the traffic coming through the gateway can back up, causing delays and incomplete transmissions for the entire internetwork. Developing an internetwork system that can dynamically adapt to loads and reroute datagram's when a bottleneck occurs is very important.

There are other factors to consider, such as network management and troubleshooting information, but you should begin to see that simply connecting networks together without due thought does not work. The many different network operating systems and hardware platforms require a logical, well-developed approach to the internetwork. This is outside the scope of TCP/IP, which is simply concerned with the transmission of the datagram's. The TCP/IP implementations on each platform, however, must be able to handle the problems mentioned.

3.8 Internet Addresses

Network addresses are analogous to mailing addresses in that they tell a system where to deliver a datagram. Three terms commonly used in the Internet relate to addressing: name, address, and route.

The term *address* is often generically used with communications protocols to refer to many different things. It can mean the destination, a port of a machine, a memory location, an application, and more. Take care when you encounter the term to make sure you know what it is really referring to.

A name is a specific identification of a machine, a user, or an application. It is usually unique and provides an absolute target for the datagram. An *address* typically identifies where the target is located, usually its physical or logical location in a network. A *route* tells the system how to get a datagram to the address.

You use the recipient's name often, either specifying a user name or a machine name, and an application does the same thing transparently to you. From the name, a network software package called the *name server* tries to resolve the address and the route, making that aspect unimportant to you. When you send electronic mail, you simply indicate the recipient's name, relying on the name server to figure out how to get the mail message to them.

Using a name server has one other primary advantage besides making the addressing and routing unimportant to the end user: It gives the system or network administrator a lot of freedom to change the network as required, without having to tell each user's machine about any changes. As long as an application can access the name server, any routing changes can be ignored by the application and users.

Naming conventions differ depending on the platform, the network, and the software release, but following is a typical Ethernet-based Internet subnetwork as an example. There are several types of addressing you need to look at, including the LAN system, as well as the wider internetwork addressing conventions.

3.8.1 Subnetwork Addressing

On a single network, several pieces of information are necessary to ensure the correct delivery of data. The primary components are the physical address and the data link address.

3.8.2 The Physical Address

Each device on a network that communicates with others has a unique physical address, sometimes called the hardware address. On any given network, there is only one occurrence of each address; otherwise, the name server has no way of identifying the target device unambiguously. For hardware, the addresses are usually encoded into a network interface card, set either by switches or by software. With respect to the OSI model, the address is located in the physical layer.

In the physical layer, the analysis of each incoming datagram (or protocol data unit) is performed. If the recipient's address matches the physical address of the device, the datagram can be passed up the layers. If the addresses don't match, the datagram is ignored. Keeping this analysis in the bottom layer of the OSI model prevents unnecessary delays, because otherwise the datagram would have to be passed up to other layers for analysis.

The length of the physical address varies depending on the networking system, but Ethernet and several others use 48 bits in each address. For communication to occur, two addresses are required: one each for the sending and receiving devices.

The IEEE is now handling the task of assigning universal physical addresses for subnetworks (a task previously performed by Xerox, as they developed Ethernet). For each subnetwork, the IEEE assigns an organization unique identifier (OUI) that is 24 bits long, enabling the organization to assign the other 24 bits however it wants. (Actually, two of the 24 bits assigned as an OUI are control bits, so only 22 bits identify the subnetwork. Because this provides 2^{22} combinations, it is possible to run out of OUIs in the future if the current rate of growth is sustained.)

The format of the OUI is shown in Figure 3.7. The least significant bit of the address (the lowest bit number) is the individual or group address bit. If the bit is set to 0, the address refers to an individual address; a setting of 1 means that the rest of the address field identifies a group address that needs further resolution. If the entire OUI is set to 1s, the address has a special meaning which is that all stations on the network are assumed to be the destination.

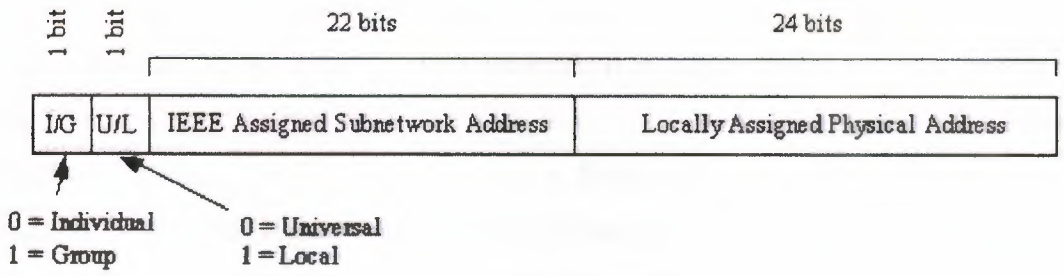


Figure 3.7. Layout of the Organization Unique Identifier.

The second bit is the *local* or *universal* bit. If set to zero, it has been set by the universal administration body. This is the setting for IEEE-assigned OUIs. If it has a value of 1, the OUI has been locally assigned and would cause addressing problems if decoded as an IEEE-assigned address.

The remaining 22 bits make up the physical address of the subnetwork, as assigned by the IEEE. The second set of 24 bits identifies local network addresses and is administered locally. If an organization runs out of physical addresses (there are about 16 million addresses possible from 24 bits), the IEEE has the capacity to assign a second subnetwork address.

The combination of 24 bits from the OUI and 24 locally assigned bits is called a media access control (MAC) address. When a packet of data is assembled for transfer across an internetwork, there are two sets of MACs: one from the sending machine and one for the receiving machine.

3.8.3 The Data Link Address

The IEEE Ethernet standards (and several other allied standards) use another address called the link layer address (abbreviated as LSAP for link service access point). The LSAP identifies the type of link protocol used in the data link layer. As with the physical address, a datagram carries both sending and receiving LSAPs. The IEEE also enables a code that identifies the Ether Type assignment, which identifies the upper layer protocol (ULP) running on the network (almost always a LAN).

3.8.4 Ethernet Frames

The layout of information in each transmitted packet of data differs depending on the protocol, but it is helpful to examine one to see how the addresses and related

information are prepended to the data. This section uses the Ethernet system as an example because of its wide use with TCP/IP. It is quite similar to other systems as well.

A typical Ethernet frame (remember that a frame is the term for a network-ready datagram) is shown in Figure 3.8. The preamble is a set of bits that are used primarily to synchronize the communication process and account for any random noise in the first few bits that are sent. At the end of the preamble is a sequence of bits that are the start frame delimiter (SFD), which indicates that the frame follows immediately.

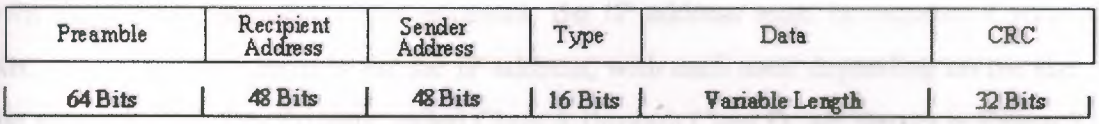


Figure 3.8. The Ethernet Frame.

The recipient and sender addresses follow in IEEE 48-bit format, followed by a 16-bit type indicator that is used to identify the protocol. The data follows the type indicator. The Data field is between 46 and 1,500 bytes in length. If the data is less than 46 bytes, it is padded with 0s until it is 46 bytes long. Any padding is not counted in the calculations of the data field's total length, which is used in one part of the IP header. The next chapter covers IP headers.

At the end of the frame is the cyclic redundancy check (CRC) count, which is used to ensure that the frame's contents have not been modified during the transmission process. Each gateway along the transmission route calculates a CRC value for the frame and compares it to the value at the end of the frame. If the two match, the frame can be sent farther along the network or into the subnetwork. If they differ, a modification to the frame must have occurred, and the frame is discarded (to be later retransmitted by the sending machine when a timer expires).

In some protocols, such as the IEEE 802.3, the overall layout of the frame is the same, with slight variations in the contents. With 802.3, the 16 bits used by Ethernet to identify the protocol type are replaced with a 16-bit value for the length of the data block. Also, the data area itself is prepended by a new field.

3.9 IP Addresses

TCP/IP uses a 32-bit address to identify a machine on a network and the network to which it is attached. IP addresses identify a machine's connection to the network, not the machine itself an important distinction. Whenever a machine's location on the network changes, the IP address must be changed, too. The IP address is the set of numbers many people see on their workstations or terminals, such as 127.40.8.72, which uniquely identifies the device.

IP (or Internet) addresses are assigned only by the Network Information Center (NIC), although if a network is not connected to the Internet, that network can determine its own numbering. For all Internet accesses, the IP address must be registered with the NIC. There are four formats for the IP address, with each used depending on the size of the network. The four formats, called Class A through Class D, are shown in Figure 3.9. The class is identified by the first few bit sequences, shown in the figure as one bit for Class A and up to four bits for Class D. The class can be determined from the first three (high-order) bits. In fact, in most cases, the first two bits are enough, because there are few Class D networks.

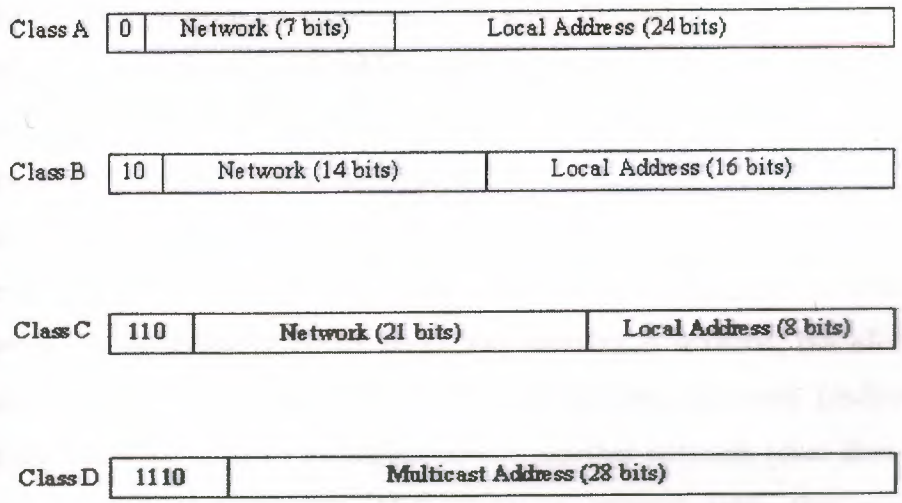


Figure 3.9. The four IP Address Class Structures.

Class A addresses are for large networks that have many machines. The 24 bits for the local address (also frequently called the host address) are needed in these cases. The

network address is kept to 7 bits, which limits the number of networks that can be identified. Class B addresses are for intermediate networks, with 16-bit local or host addresses and 14-bit network addresses. Class C networks have only 8 bits for the local or host address, limiting the number of devices to 256. There are 21 bits for the network address. Finally, Class D networks are used for multicasting purposes, when a general broadcast to more than one device is required. The lengths of each section of the IP address have been carefully chosen to provide maximum flexibility in assigning both network and local addresses.

IP addresses are four sets of 8 bits, for a total 32 bits. You often represent these bits as separated by a period for convenience, so the IP address format can be thought of as network.local.local.local for Class A or network.network.network.local for Class C. The IP addresses are usually written out in their decimal equivalents, instead of the long binary strings. This is the familiar host address number that network users are used to seeing, such as 147.10.13.28, which would indicate that the network address is 147.10 and the local or host address is 13.28. Of course, the actual address is a set of 1s and 0s. The decimal notation used for IP addresses is properly called *dotted quad notation*—a bit of trivia for your next dinner party.

The IP addresses can be translated to common names and letters. This can pose a problem, though, because there must be some method of unambiguously relating the physical address, the network address, and a language-based name (such a tpci_ws_4 or bobs machine). The section later in this chapter titled "The Domain Name System" looks at this aspect of address naming.

From the IP address, a network can determine if the data is to be sent out through a gateway. If the network address is the same as the current address (routing to a local network device, called a *direct host*), the gateway is avoided, but all other network addresses are routed to a gateway to leave the local network (*indirect host*). The gateway receiving data to be transmitted to another network must then determine the routing from the data's IP address and an internal table that provides routing information.

As mentioned, if an address is set to all 1s, the address applies to all addresses on the network. (See the previous section titled "Physical Addresses.") The same rule applies to IP addresses, so that an IP address of 32 1s is considered a broadcast message to all networks and all devices. It is possible to broadcast to all machines in a network by

altering the local or host address to all 1s, so that the address 147.10.255.255 for a Class B network (identified as network 147.10) would be received by all devices on that network (255.255 being the local addresses composed of all 1s), but the data would not leave the network.

There are two contradictory ways to indicate broadcasts. The later versions of TCP/IP use 1s, but earlier BSD systems use 0s. This causes a lot of confusion. All the devices on a network must know which broadcast convention is used; otherwise, datagram's can be stuck on the network forever!

A slight twist is coding the network address as all 0s, which means the originating network or the local address being set to 0s, which refers to the originating device only (usually used only when a device is trying to determine its IP address). The all-zero network address format is used when the network IP address is not known but other devices on the network can still interpret the local address. If this were transmitted to another network, it could cause confusion! By convention, no local device is given a physical address of 0.

It is possible for a device to have more than one IP address if it is connected to more than one network, as is the case with gateways. These devices are called *multihomed,* because they have a unique address for each network they are connected to. In practice, it is best to have a dedicate machine for a multihomed gateway; otherwise, the applications on that machine can get confused as to which address they should use when building datagram's.

Two networks can have the same network address if they are connected by a gateway. This can cause problems for addressing; because the gateway must be able to differentiate which network the physical address is on. This problem is looked at again in the next section, showing how it can be solved.

3.10 Address Resolution Protocol

Determining addresses can be difficult because every machine on the network might not have a list of all the addresses of the other machines or devices. Sending data from one machine to another if the recipient machine's physical address is not known can cause a problem if there is no resolution system for determining the addresses. Having to constantly update a table of addresses on each machine would be a network administration nightmare. The problem is not restricted to machine addresses within a

small network, because if the remote destination network addresses are unknown, routing and delivery problems will also occur.

The Address Resolution Protocol (ARP) helps solve these problems. ARP's job is to convert IP addresses to physical addresses (network and local) and in doing so, eliminate the need for applications to know about the physical addresses. Essentially, ARP is a table with a list of the IP addresses and their corresponding physical addresses. The table is called an *ARP cache.* The layout of an ARP cache is shown in Figure 3.10. Each row corresponds to one device, with four pieces of information for each device:

	IF INDEX	PHYSICAL ADDRESS	IP ADDRESS	TYPE
Entry 1				
Entry 2				
Entry 3				
Entry n				

Figure 3.10. The ARP Cache Address Translation Table Layout.

- IF Index: The physical port (interface)
- Physical Address: The physical address of the device
- IP Address: The IP address corresponding to the physical address
- Type: The type of entry in the ARP cache

3.10.1 Mapping Types

The mapping type is one of four possible values indicating the status of the entry in the ARP cache. A value of 2 means the entry is invalid; a value of 3 means the

mapping is dynamic (the entry can change); a value of 4 means static (the entry doesn't change); and a value of 1 means none of the above.

When the ARP receives a recipient device's IP address, it searches the ARP cache for a match. If it finds one, it returns the physical address. If the ARP cache doesn't find a match for an IP address, it sends a message out on the network. The message, called an *ARP request,* is a broadcast that is received by all devices on the local network. (You might remember that a broadcast has all 1s in the address.) The ARP request contains the IP address of the intended recipient device. If a device recognizes the IP address as belonging to it, the device sends a reply message containing its physical address back to the machine that generated the ARP request, which places the information into its ARP cache for future use. In this manner, the ARP cache can determine the physical address for any machine based on its IP address.

Whenever an ARP request is received by an ARP cache, it uses the information in the request to update its own table. Thus, the system can accommodate changing physical addresses and new additions to the network dynamically without having to generate an ARP request of its own. Without the use of an ARP cache, all the ARP requests and replies would generate a lot of network traffic, which can have a serious impact on network performance. Some simpler network schemes abandon the cache and simply use broadcast messages each time. This is feasible only when the number of devices is low enough to avoid network traffic problems.

The layout of the ARP request is shown in Figure 3.11. When an ARP request is sent, all fields in the layout are used except the Recipient Hardware Address (which the request is trying to identify). In an ARP reply, all the fields are used.

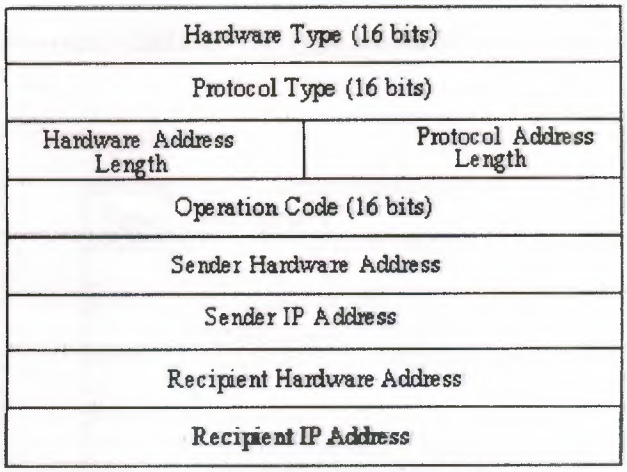


Figure 3.11. The ARP Request and ARP Reply Layout.

This layout, which is combined with the network system's protocols into a protocol data unit (PDU), has several fields. The fields and their purposes are as follows:

- Hardware Type: The type of hardware interface
- Protocol Type: The type of protocol the sending device is using
- Hardware Address Length: The length of each hardware address in the datagram, given in bytes
- Protocol Address Length: The length of the protocol address in the datagram, given in bytes
- Operation Code (Opcode): The Opcode indicates whether the datagram is an ARP request or an ARP reply. If the datagram is a request, the value is set to 1. If it is a reply, the value is set to 2.
- Sender Hardware Address: The hardware address of the sending device
- Sender IP Address: The IP address of the sending device
- Recipient IP Address: The IP Address of the recipient
- Recipient Hardware Address: The hardware address of the recipient device

Some of these fields need a little more explanation to show their legal values and field usage. The following sections describe these fields in more detail.

3.10.2 The Hardware Type Field

The hardware type identifies the type of hardware interface. Legal values are as follows:

Type	*Description*
1	Ethernet
2	Experimental Ethernet
3	X.25
4	Proteon ProNET (Token Ring)
5	Chaos
6	IEEE 802.X
7	ARCnet

3.10.3 The Protocol Type Field

The protocol type identifies the type of protocol the sending device is using. With TCP/IP, these protocols are usually an EtherType, for which the legal values are as follows:

Decimal	*Description*
512	XEROX PUP
513	PUP Address Translation
1536	XEROX NS IDP
2048	Internet Protocol (IP)
2049	X.75
2050	NBS
2051	ECMA
2052	Chaosnet
2053	X.25 Level 3
2054	Address Resolution Protocol (ARP)
2055	XNS
4096	Berkeley Trailer
21000	BBN Simnet

24577	DEC MOP Dump/Load
24578	DEC MOP Remote Console
24579	DEC DECnet Phase IV
24580	DEC LAT
24582	DEC
24583	DEC
32773	HP Probe
32784	Excelan
32821	Reverse ARP
32824	DEC LANBridge
32823	AppleTalk

If the protocol is not Ether Type, other values are allowed.

3.11 ARP and IP Addresses

Two (or more) networks connected by a gateway can have the same network address. The gateway has to determine which network the physical address or IP address corresponds with. The gateway can do this with a modified ARP, called the Proxy ARP (sometimes called Promiscuous ARP). A proxy ARP creates an ARP cache consisting of entries from both networks, with the gateway able to transfer datagrams from one network to the other. The gateway has to manage the ARP requests and replies that cross the two networks.

An obvious flaw with the ARP system is that if a device doesn't know its own IP address, there is no way to generate requests and replies. This can happen when a new device (typically a diskless workstation) is added to the network. The only address the device is aware of is the physical address set either by switches on the network interface or by software. A simple solution is the Reverse Address Resolution Protocol (RARP), which works the reverse of ARP, sending out the physical address and expecting back an IP address. The reply containing the IP address is sent by an RARP server, a machine that can supply the information. Although the originating device sends the message as a broadcast, RARP rules stipulate that only the RARP server can generate a reply. (Many networks assign more than one RARP server, both to spread the processing load and to act as a backup in case of problems.)

3.12 The Domain Name System

Instead of using the full 32-bit IP address, many systems adopt more meaningful names for their devices and networks. Network names usually reflect the organization's name (such as tpci.com and bobs_cement). Individual device names within a network can range from descriptive names on small networks (such as tims_machine and laser_1) to more complex naming conventions on larger networks (such as hpws_23 and tpci704). Translating between these names and the IP addresses would be practically impossible on an Internet-wide scale.

To solve the problem of network names, the Network Information Center (NIC) maintains a list of network names and the corresponding network gateway addresses. This system grew from a simple flat-file list (which was searched for matches) to a more complicated system called the Domain Name System (DNS) when the networks became too numerous for the flat-file system to function efficiently.

DNS uses a hierarchical architecture, much like the UNIX filesystem. The first level of naming divides networks into the category of subnetworks, such as com for commercial, mil for military, edu for education, and so on. Below each of these is another division that identifies the individual subnetwork, usually one for each organization. This is called the *domain name* and is unique. The organization's system manager can further divide the company's subnetworks as desired, with each network called a *subdomain.* For example, the system merlin.abc_corp.com has the domain name abc_corp.com, whereas the network merlin.abc_corp is a subdomain of merlin.abc_corp.com. A network can be identified with an *absolute name* (such as merlin.abc_corp.com) or a *relative name* (such as merlin) that uses part of the complete domain name.

Seven first-level domain names have been established by the NIC so far. These are as follows:

.arpa	An ARPANET-Internet identification
.com	Commercial company
.edu	Educational institution
.gov	Any governmental body
.mil	Military
.net	Networks used by Internet Service Providers

.org	Anything that doesn't fall into one of the other categories

The NIC also allows for a country designator to be appended. There are designators for all countries in the world, such as .ca for Canada and .uk for the United Kingdom.

DNS uses two systems to establish and track domain names. A *name resolver* on each network examines information in a domain name. If it can't find the full IP address, it queries a *name server,* which has the full NIC information available. The name resolver tries to complete the addressing information using its own database, which it updates in much the same manner as the ARP system (discussed earlier) when it must query a name server. If a queried name server cannot resolve the address, it can query another name server, and so on, across the entire internetwork.

There is a considerable amount of information stored in the name resolver and name server, as well as a whole set of protocols for querying between the two. The details, luckily, are not important to an understanding of TCP/IP, although the overall concept of the address resolution is important when understanding how the Internet translates between domain names and IP addresses.

3.13 Summary

In this chapter you have seen the relationship of OSI and TCP/IP layered architectures, a history of TCP/IP and the Internet, the structure of the Internet, Internet and IP addresses, and the Address Resolution Protocol. Using these concepts, you can now move on to look at the TCP/IP family of protocols in more detail.

CHAPTER FOUR
A CHAT CLIENT-SERVER MODULE IN JAVA

4.1 Overview

This Java program implements user-friendly interface of Client/Server architectures, a particularly powerful and flexible design in which a process running on one machine can respond to requests for data and/or services from programs running on different machines.

In our program, the Client is running on the user's local PC, and the Server is running on the user's local PC. Once a socket connection request is submitted from Client/Server interface, the Server will first be connect at port 7888 on blackbird, if this port is in use, the user may change the port number to make a connection.

4.2 Transmission Control Protocol/Internet Protocol (TCP/IP)

This protocol ensures that the data sent over two points in a Network is received in the same order as it is sent.

4.2.1 Port Number

Is an address, which determines the data origin and delivery points over the Network?

4.2.2 Sockets

A socket is one-end point of a two-way communication link between two programs running on the Network.

4.3 The Java.net Package

The java.net package is used as the basis for this program. It provides two classes:

- **Socket**
- **Server Socket**

The Socket class implements the client side of the connection and the Server Socket Implements the server side of the connection.

- **On the server side:**

  Normally a server runs on a specific computer and has a socket bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

- **On the client side:**

  The client knows the host name of the machine on which the server is running and the port number to which the server is connected. To make a connection request, the client tries to communicate with the server on the server's machine and port.

4.4 Source Code

4.4.1 Supporting Multiple Clients

Client connection requests are queued at the port, so the server must accept the connections sequentially. However, the server can service them simultaneously. Through the use of threads, one thread per client connection.

4.4.2 A Brief Description of The Various Files in The Source Code

- **Server Section:**

  1. **Client List**

     The client list class creates a client list object, which stores a list of output streams connected to individual clients.

  2. **Session**

     This class creates a session object, which listens to the client for messages. It keeps listening until it receives a "Connect" message. Currently, once the session receives a "Connect" message, it generates a new message saying that user has logged on the first thing a session object does is add the output stream which leads to the client into the client list object. This means that any input sent from any client will be redirected through the whole list.

  3. **Server:**

     The Server Class creates a server object, instantiates a new server listener, opens a new server socket, launches a new thread and listens to it. Whenever a new client connects, the server object creates a new session object for the client and goes back to listening.

- **Client Section:**

1. **Client:**

   The Client object opens a socket to the server and creates a user interface for the user once everything has been initialized it launches a Client Listener object (running on a Separate thread) which listens to the server and informs the client when new text Arrives. The client object then waits for an action event, and sends any text typed by the user back to the server.

2. **Message:**

   This is a simple class, which describes a message passed between a server and a client.

3. **Client Listener:**

   This class creates a new thread, which listens to the server for any messages and passes them back to the client.

4.5 Explanation of a Chat Client/Server Program by Block Diagram

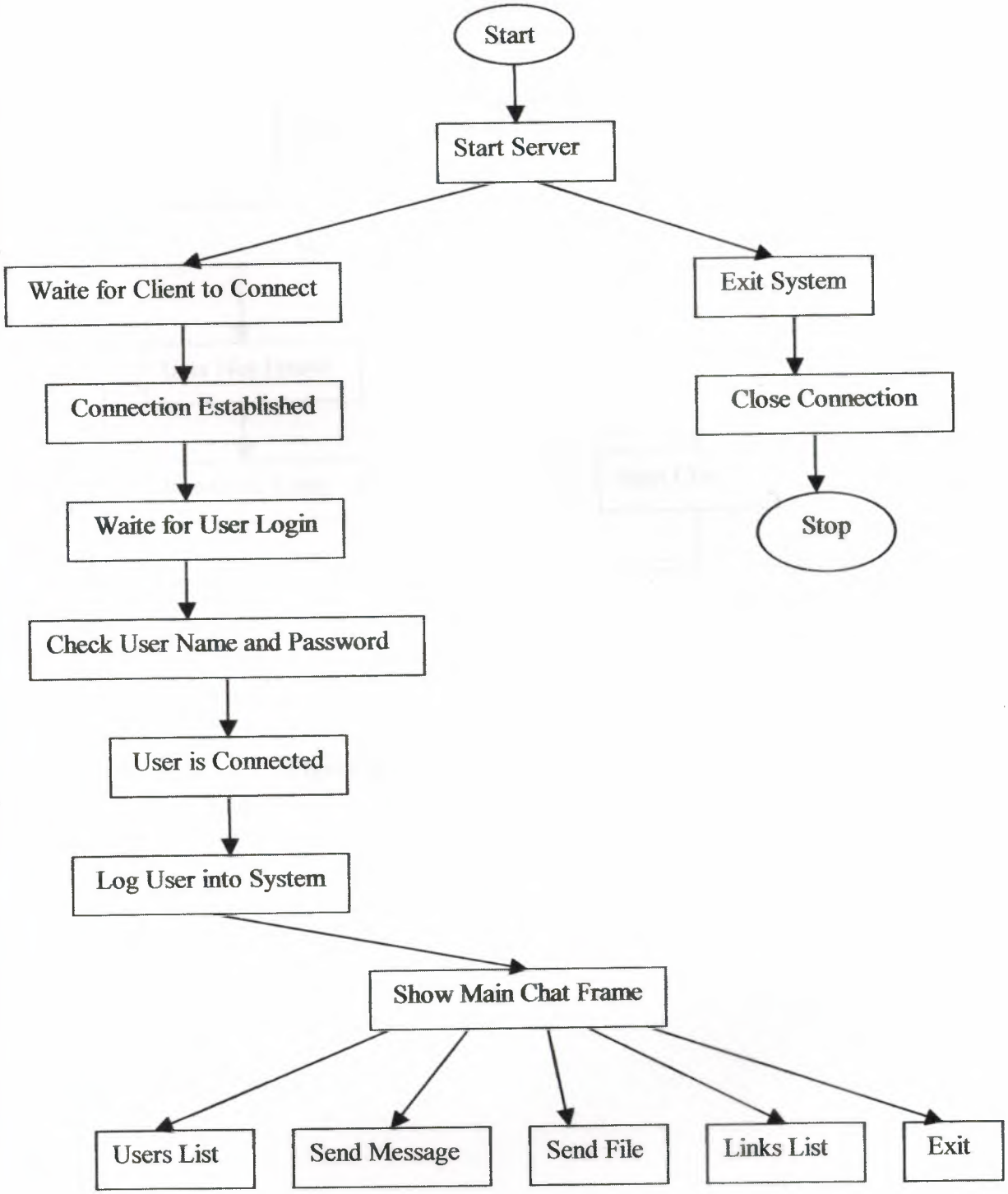


Figure 4.1 Client/Server Block Diagram.

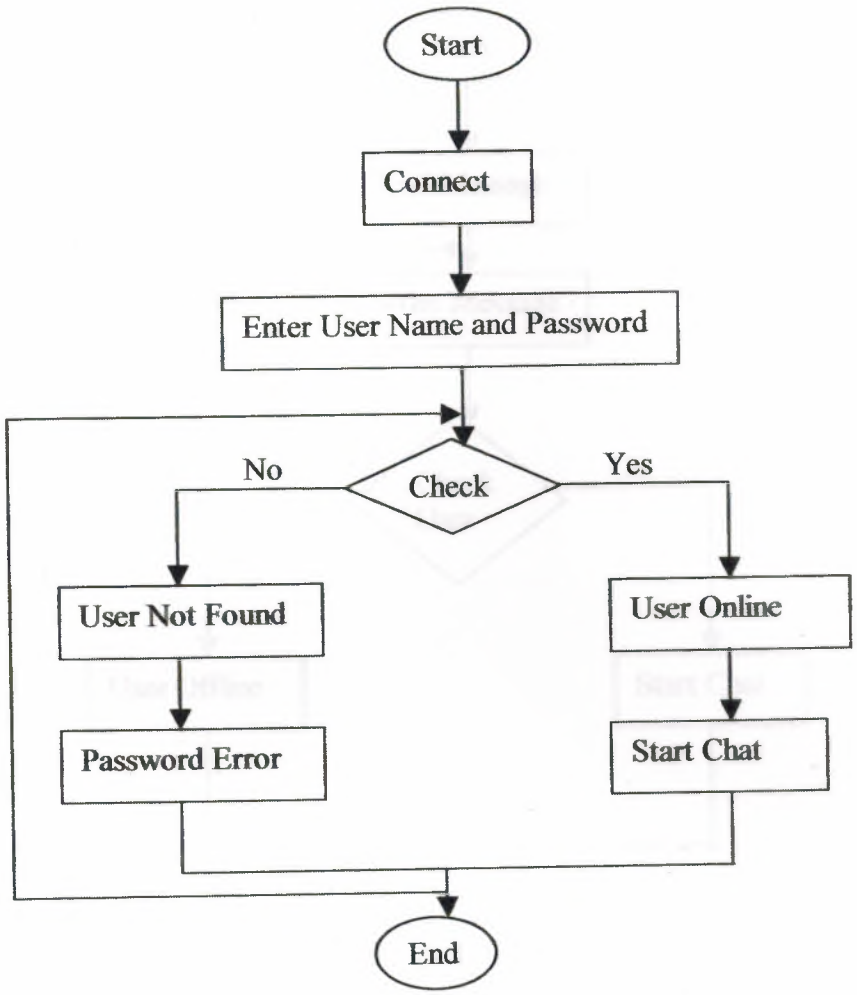


Figure 4.2 Connection Flow Chart.

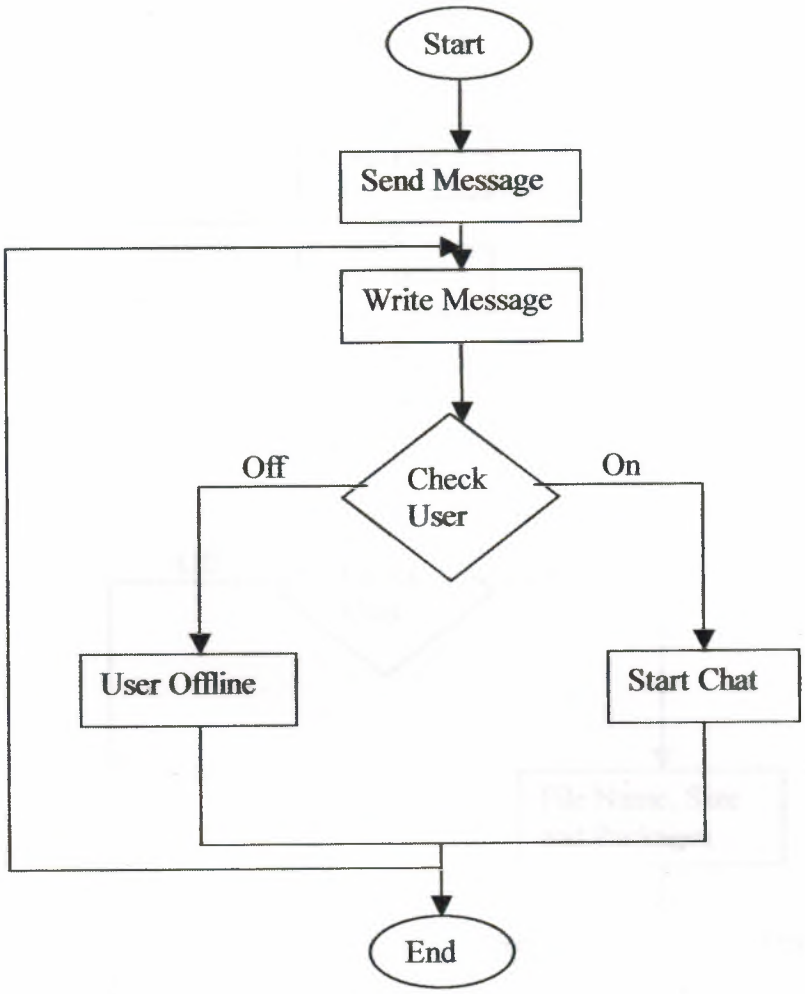


Figure 4.3 Send Message Flow Chart.

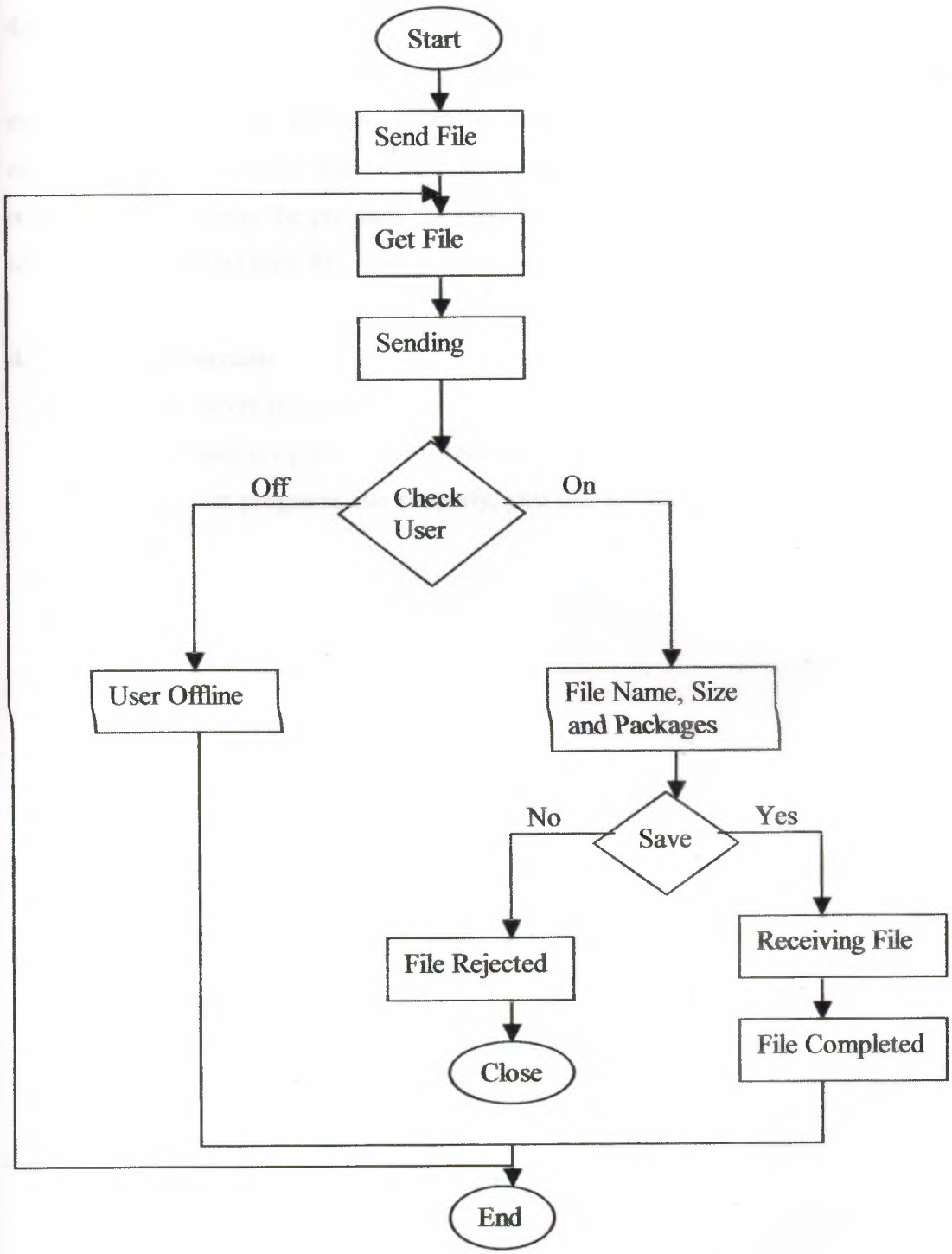


Figure 4.4 Send File Flow Chart.

4.6 Program Objectives

The main objective of the application is to create a Chat Client/Server environment. It allows multiple users to login by entering user name, password, connects to server, convey text-formatted information, logoff and re-login any time they want to. For the server the program can be run on local PC. The client program also can be run stand-alone on local PC. This gives this application great flexible usage.

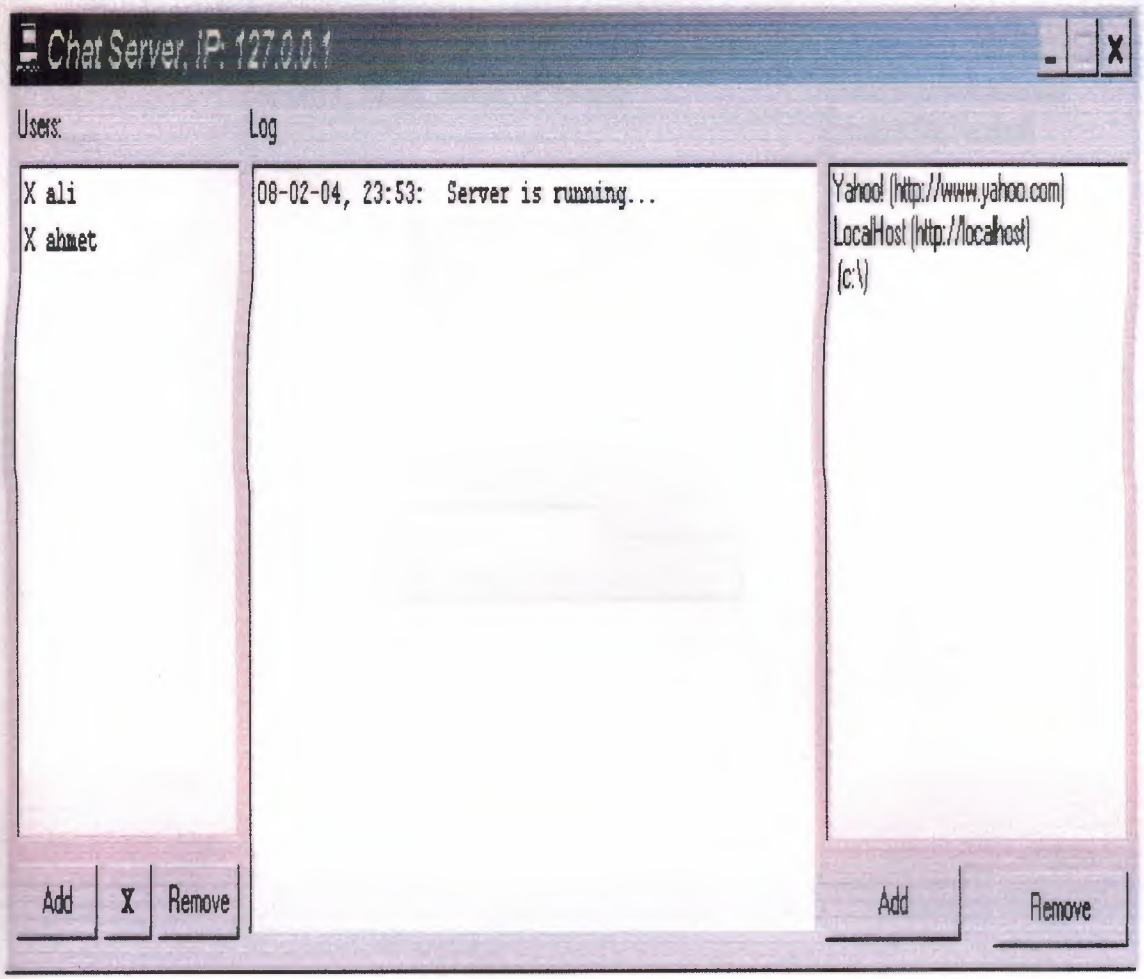
4.7 Run The Programs

- Start the server program on one machine.
- Run the client program on any other machine. Input the server host IP number.
- Then if your programs run properly, you can see the messages appearing on the screen.

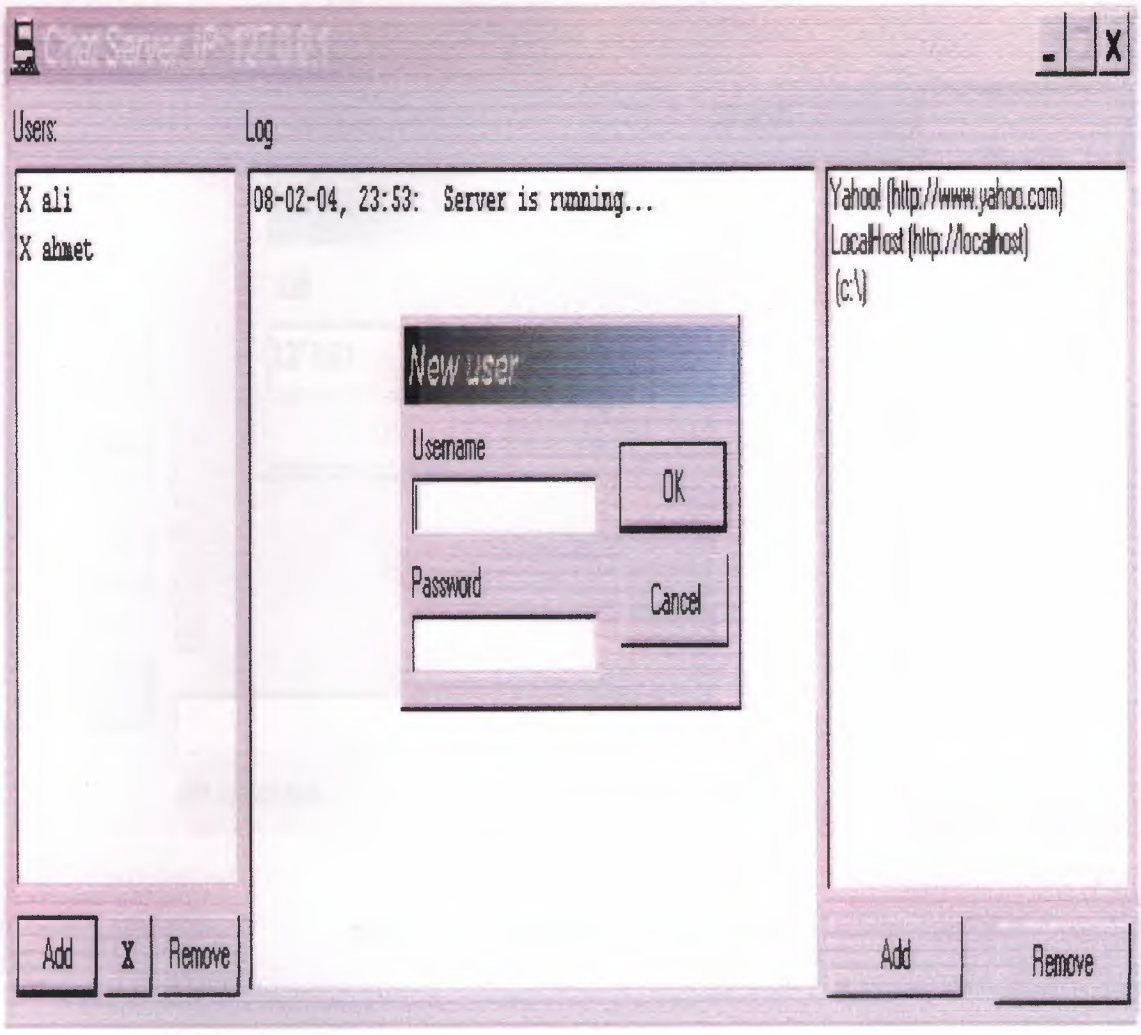
4.8 Summary

This was the most important chapter of this project, which describes the chat Client/Server application in details. I developed distributed client/server chat applications in the Java language. It has capability to manage multiple users sending or receiving information simultaneously. It provides graphic user interface, fast communication, and secure Client/Server environment, and includes an explanation of the program, application objectives and run the programs, also block diagram was provided in this chapter, and the program is given in an appendix at the end of the project.

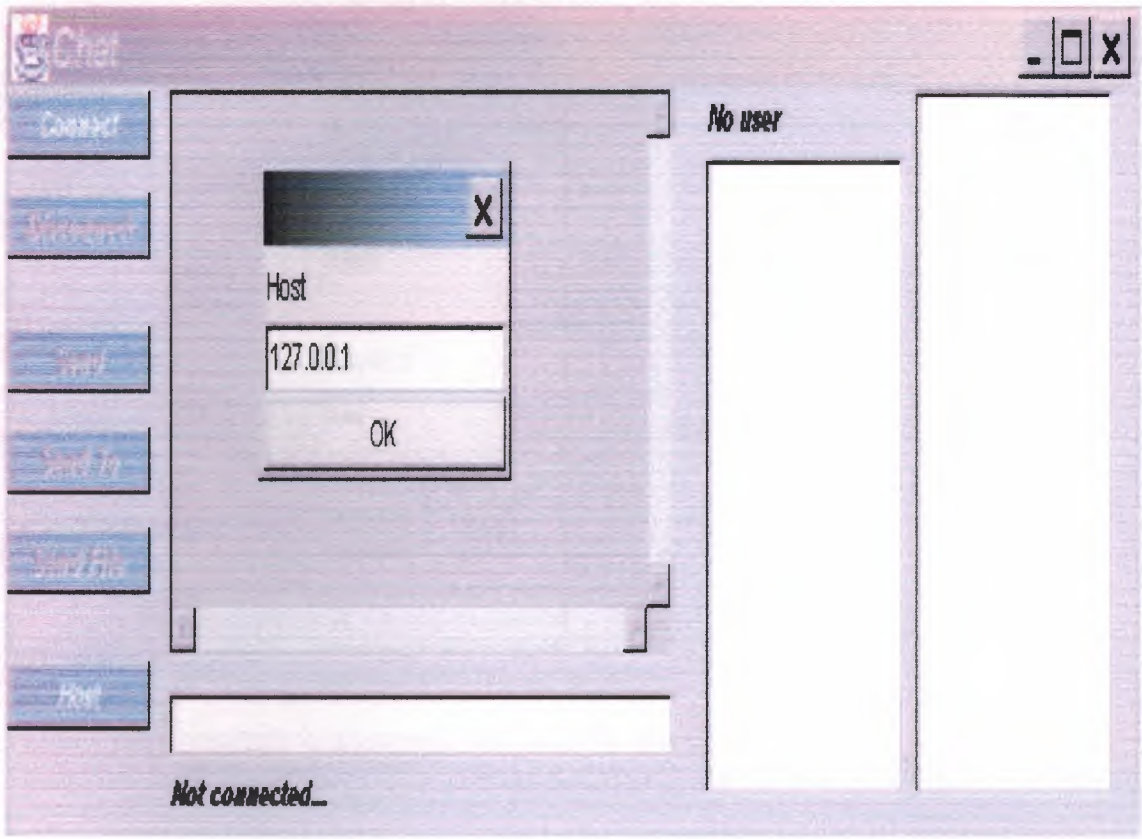
The Screen Layout of the Program



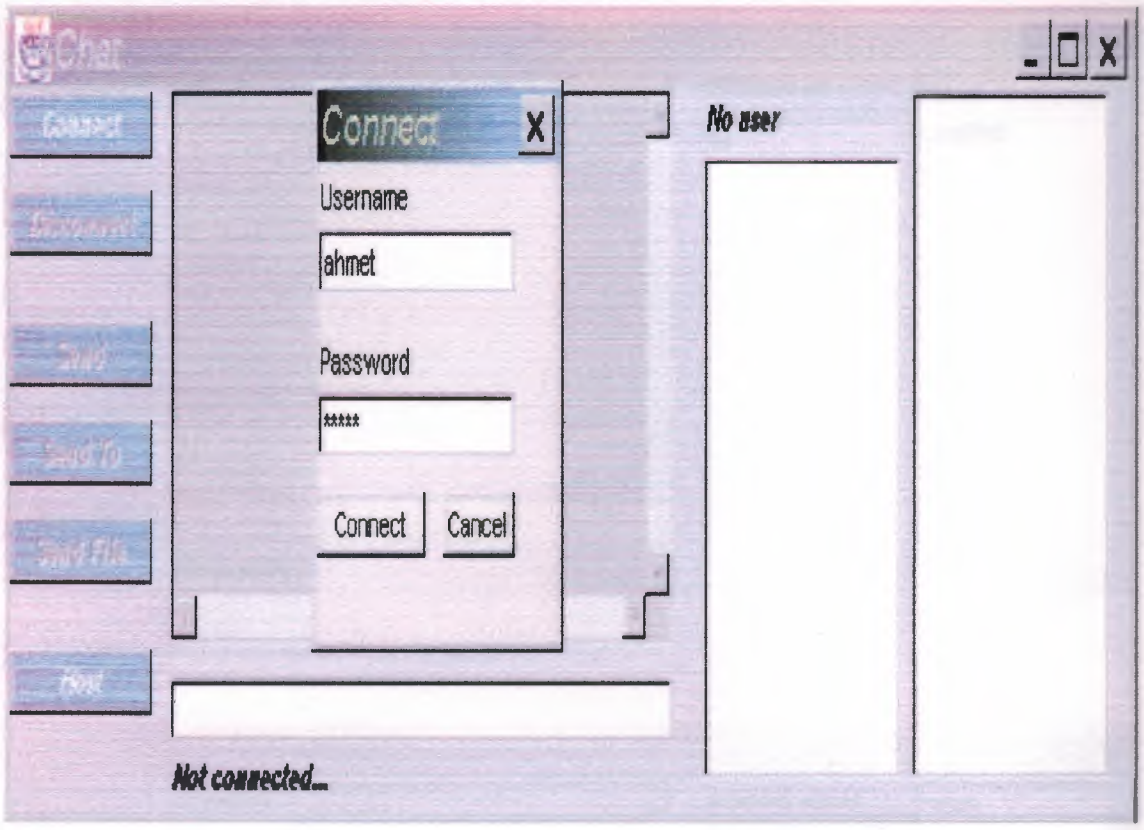
Server Application Started


Users:
Log
X ali
X ahmet
08-02-04, 23:53: Server is running...
Yahoo! (http://www.yahoo.com)
LocalHost (http://localhost)
(c:\)
New user
Username
OK
Password
Cancel
Add
X
Remove
Add
Remove

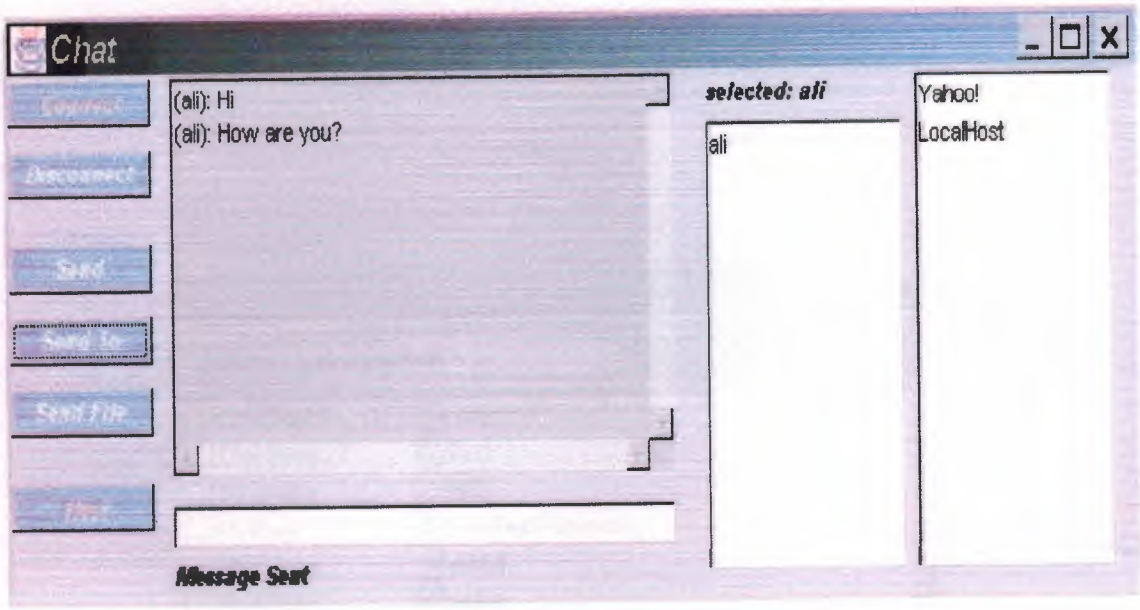
Add New User

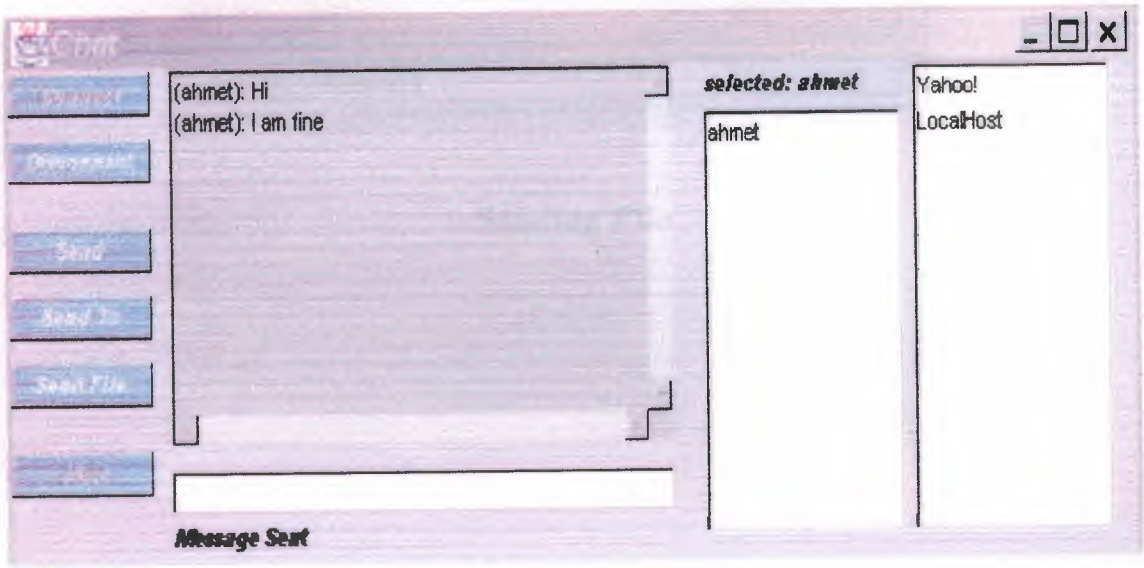


Start Client and Setting Host

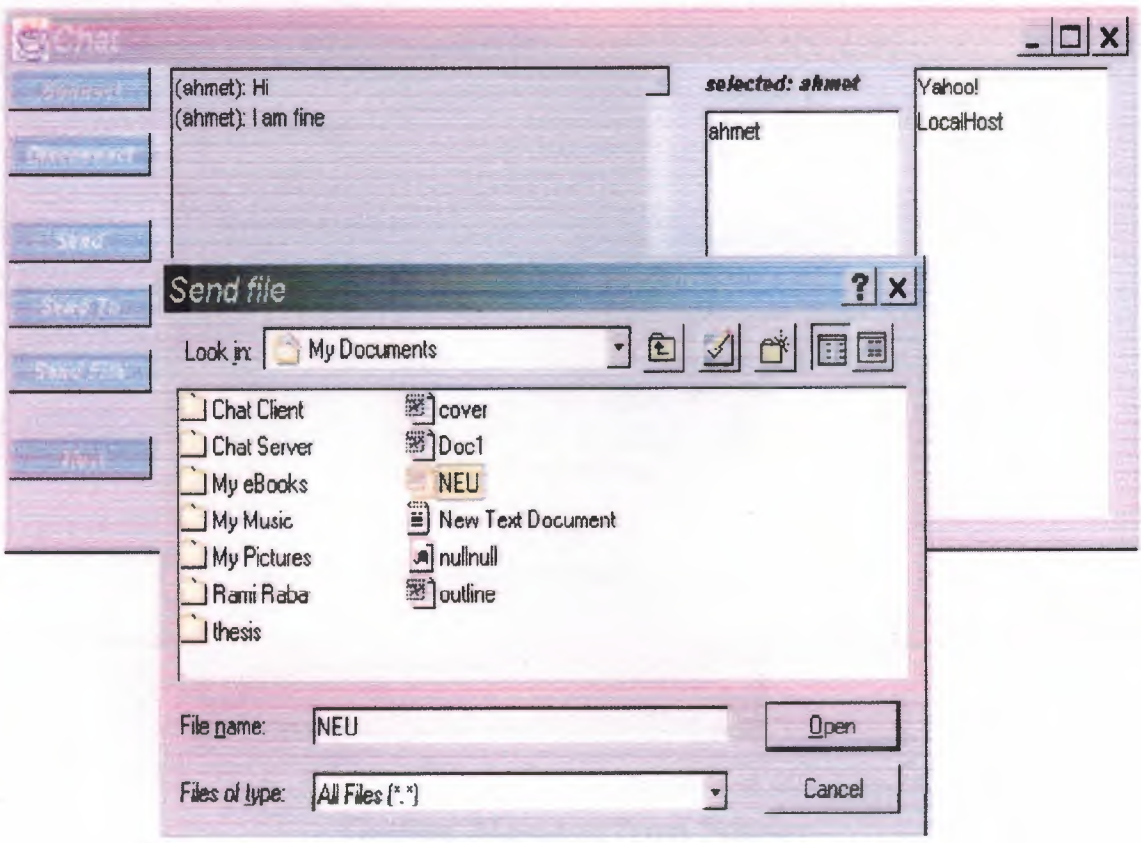


Connecting User with Server

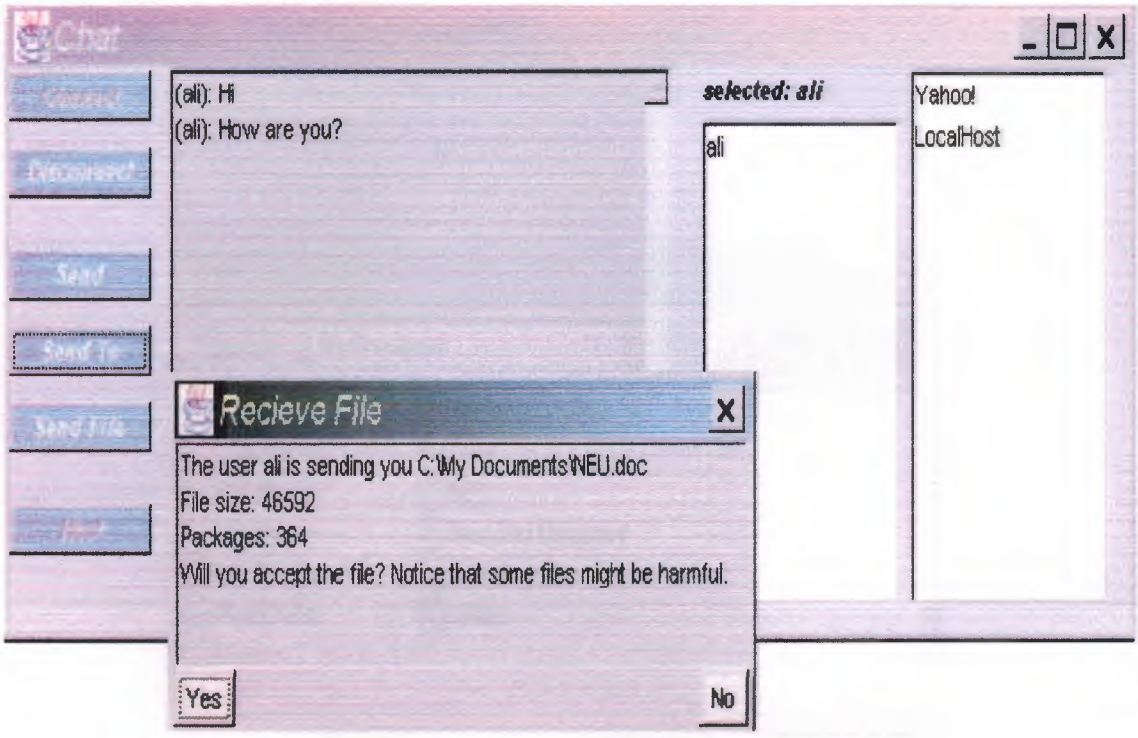




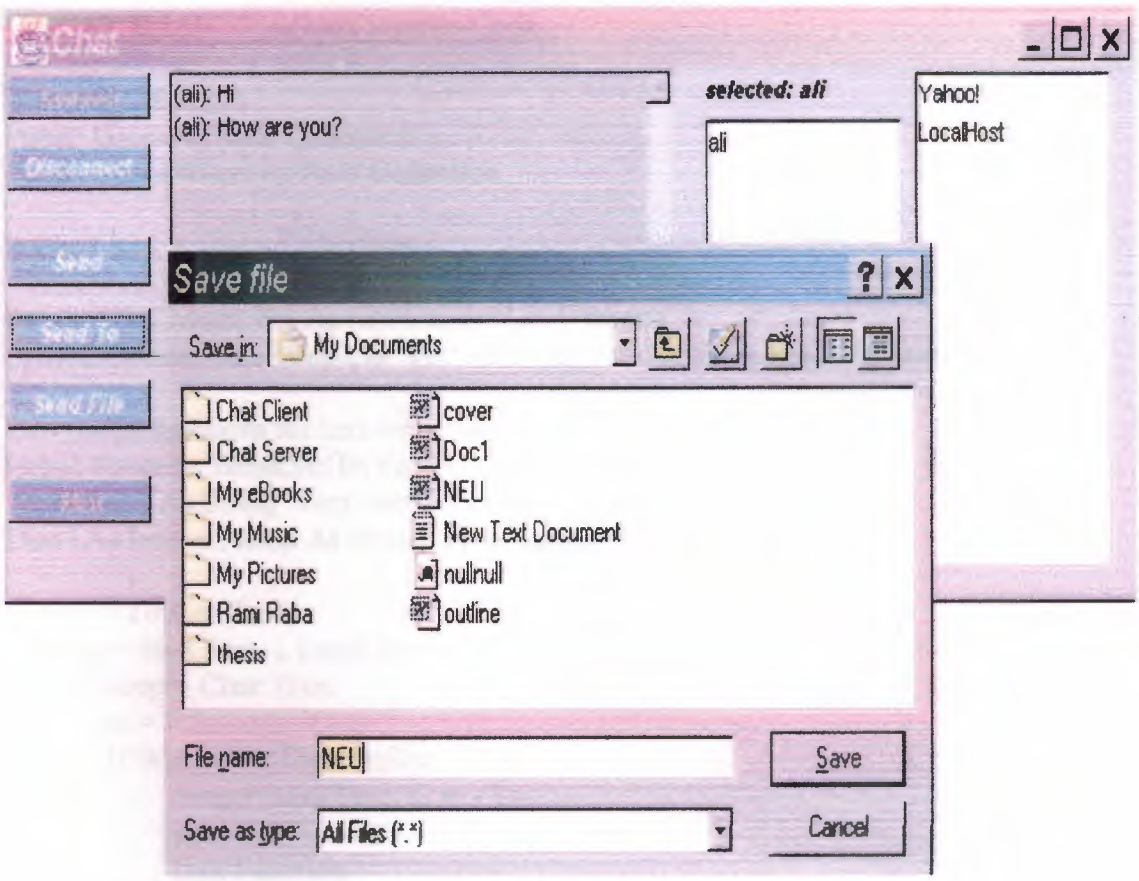
Start Chatting Between Users



Sending File



Getting Message from Sender



Receiver Saving File

VISUAL BASIC Program Source-Code

```
'our user collection
Public Users As New Collection
Public Connections As New Collection

Public Links As New Collection

Public WithEvents WS As MSWinsockLib.Winsock
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'two useful functions for text manipulation
Public Function BreakStr(ByVal Text As String, ByVal Char As String, ByVal Where
As Integer) As String 'Very useful function
Dim i As Integer, temp As String, m As Integer

For i = 1 To Len(Text)
   temp = Mid(Text, i, Len(Char))
      If temp = Char Then
         m = i
         If Where = 1 Then 'before
               temp = Mid(Text, 1, m - 1)
               BreakStr = temp
               Exit Function
         Else
               temp = Mid(Text, m + Len(Char), Len(Text))
               BreakStr = temp
               Exit Function
         End If
      End If
Next
If Where = 1 Then BreakStr = Text
End Function

Public Function BreakStrEx(ByVal Text As String, ByVal Char As String, ByVal
Where As Integer) As String
Dim i As Integer, temp As String, m As Integer

For i = Len(Text) To 1 Step -1
   temp = Mid(Text, i, Len(Char))
      If temp = Char Then
         m = i
         If Where = 1 Then 'before
               temp = Mid(Text, 1, m - Len(Char))
               BreakStrEx = temp
               Exit Function
         Else
               temp = Mid(Text, m + Len(Char), Len(Text))
               BreakStrEx = temp
```

```
                Exit Function
            End If
        End If
    Next
    If Where = 0 Then BreakStrEx = Text
    End Function

    Private Sub cmdAdd_Click()
    frmNUsr.Show , Me
    End Sub

    Private Sub cmdLAdd_Click()
    frmNLnk.Show , Me
    End Sub

    Private Sub cmdLRem_Click()
    On Error GoTo ErrH
    ret = MsgBox("Are you sure to delete " & Links(lstLinks.ListIndex + 1).Caption,
    vbYesNo, "Remove")
    If ret = vbNo Then Exit Sub
    Links.Remove lstLinks.ListIndex + 1

    Open App.Path & "\links.dat" For Output As #1
    For i = 1 To frmServer.Links.Count
       Print #1, frmServer.Links(i).Caption
       Print #1, frmServer.Links(i).Link
    Next
    Close
    frmServer.RefreshList

    Exit Sub
    ErrH:
       log Err & Chr(9) & Err.Description
    End Sub

    Private Sub cmdOffline_Click()
    On Error GoTo ErrH
    Dim Conn As clsConnection

    Set Conn = Users(lstOnline.ListIndex + 1).Connection
    If Not Conn Is Nothing Then
       Conn.GetWS.Close
       Conn.State = notConnected
       Connections.Remove Conn.ID
       Set Users(lstOnline.ListIndex + 1).Connection = Nothing
    End If

    Users(lstOnline.ListIndex + 1).isOnline = False
    log Users(lstOnline.ListIndex + 1).UserName & " is kicked"
```

```
RefreshList

Exit Sub
ErrH:
   log Err & Chr(9) & Err.Description
End Sub

Private Sub cmdRemove_Click()
On Error GoTo ErrH
ret = MsgBox("Are you sure to delete " & Users(lstOnline.ListIndex + 1).UserName,
vbYesNo, "Remove")
If ret = vbNo Then Exit Sub
Users.Remove lstOnline.ListIndex + 1

Open App.Path & "\users.dat" For Output As #1
For i = 1 To frmServer.Users.Count
   Print #1, frmServer.Users(i).UserName
   Print #1, frmServer.Users(i).Password
Next
Close
frmServer.RefreshList

Exit Sub
ErrH:
   log Err & Chr(9) & Err.Description
End Sub

'start listening oaur port
Private Sub Form_Load()
On Error GoTo ErrH
Dim usrcls As clsUser
Dim lnkcls As clsLink
'generate user list
'open user file
Open App.Path & "\users.dat" For Input As #1
Do Until EOF(1)
   'get user name
   Line Input #1, usr
   'get user password
   Line Input #1, pass
   'create a new class
   Set usrcls = New clsUser
   'set properties
   usrcls.UserName = usr
   usrcls.Password = pass
   'add to collection
   Users.Add usrcls
Loop
'close file
```

```
Close

Open App.Path & "\links.dat" For Input As #1
Do Until EOF(1)
   Line Input #1, cap
   Line Input #1, lnk
   Set lnkcls = New clsLink
   lnkcls.Caption = cap
   lnkcls.Link = lnk

   Links.Add lnkcls
Loop
Close

'refresh user list
RefreshList

Set WS = New Winsock
WS.LocalPort = 7888
WS.Listen
log "Server is running..."

Exit Sub
ErrH:
   log Err & Chr(9) & Err.Description
End Sub

'refresh users list
Sub RefreshList()
On Error GoTo ErrH
'clear list
lstOnline.Clear
lstLinks.Clear
'add users to list
usrs = ""
usrc = 0
For i = 1 To Users.Count
   lstOnline.AddItem IIf(Users(i).isOnline, "O ", "X ") & Users(i).UserName
   If Users(i).isOnline Then usrc = usrc + 1
Next

lnk = ""
For i = 1 To Links.Count
   lstLinks.AddItem Links(i).Caption & " (" & Links(i).Link & ")"
   lnk = lnk & IIf(lnk = "", "", "*") & Links(i).Caption
Next

lnk = "-" & Links.Count & "*" & lnk
msg = "6" & usrc & usrs
```

```
usrc = usrc - 1
For i = 1 To Users.Count
   If Users(i).isOnline Then
      If Users(i).Connection Is Nothing Then
         Users(i).isOnline = False
      Else
         If Users(i).Connection.GetWS.State <> 7 Then
            Set Users(i).Connection = Nothing
            Users(i).isOnline = False
         Else
            usrs = ""
            For j = 1 To Users.Count
               If Users(j).isOnline And Not (j = i) Then usrs = usrs & "*" &
Users(j).UserName
            Next
            msg = "6" & usrc & usrs

            Users(i).Connection.GetWS.SendData msg
            DoEvents
            Users(i).Connection.GetWS.SendData lnk
            DoEvents
         End If
      End If
   End If
Next

Exit Sub
ErrH:
   log Err & Chr(9) & Err.Description
End Sub

'log a string
Sub log(strn)
lstLog.AddItem Format(Now, "dd-mm-yy, hh:mm") & ":  " & strn, 0
End Sub

'refresh user list
Private Sub tmrRefresh_Timer()
RefreshList
End Sub

'connection closed
Public Sub Conn_Close(Conn As clsConnection)
On Error GoTo ErrH
Dim User As New clsUser

log Conn.GetWS.RemoteHostIP & " is disconnected..."

If Not Conn.User Is Nothing Then
```

```
    Set Conn.User.Connection = Nothing
    Conn.User.isOnline = False
End If

Connections.Remove Conn.ID
Conn.State = notConnected

RefreshList
Exit Sub
ErrH:
    log Err & Chr(9) & Err.Description
End Sub

'incoming connection
Private Sub WS_ConnectionRequest(ByVal requestID As Long)
On Error GoTo ErrH
Dim Conn As New clsConnection

Conn.NewWS

'Accept connection
Conn.GetWS.Accept requestID
log Conn.GetWS.RemoteHostIP & " is connected..."

'send confirmation
Conn.GetWS.SendData "2"

Conn.State = NewConnection
Set Conn.Parent = Me

Connections.Add Conn
Conn.ID = Connections.Count
Conn.State = NewConnection

Exit Sub
ErrH:
    log Err & Chr(9) & Err.Description
End Sub

Public Sub UserDat(Conn As clsConnection, ByVal Data As String)
Dim St As Boolean
On Error GoTo ErrH
Select Case Conn.State
Case NewConnection
    St = True
    St = St And (Left(Data, 1) = "1")
    usrn = BreakStr(Mid(Data, 2), "*", 1)
    usrp = BreakStr(Mid(Data, 2), "*", 0)
    If Not St Then
```

```
            log "Data integration error with " & Conn.GetWS.RemoteHostIP
            Conn.GetWS.SendData "9"
            Conn_Close Conn
            Exit Sub
        End If
        For i = 1 To Users.Count
            If usrn = Users(i).UserName Then Exit For
        Next
        If i = Users.Count + 1 Then
            Conn.GetWS.SendData "0"
            log "User name " & usrn & " not found (ip: " & Conn.GetWS.RemoteHostIP & ")"
            Exit Sub
        End If

        If Users(i).isOnline Then
            Conn.GetWS.SendData "3"
            log usrn & " is already online"
            Exit Sub
        End If

        If Users(i).Password <> usrp Then
            Conn.GetWS.SendData "1"
            log "Password error for " & usrn & " (ip: " & Conn.GetWS.RemoteHostIP & ")"
            Exit Sub
        End If

        Conn.GetWS.SendData "2"
        log usrn & " is online at " & Conn.GetWS.RemoteHostIP
        Conn.State = Online
        Set Conn.User = Users(i)
        Set Users(i).Connection = Conn
        Users(i).isOnline = True
        RefreshList
    Case Online
        Select Case Left(Data, 1)
        Case 1
            'send integration error and close
            Conn.GetWS.SendData "9"
            Conn.State = notConnected
        Case 2
            Sleep 50
            'request recieved
            Conn.GetWS.SendData "2"
            DoEvents
            'send message to all online users
            For i = 1 To Users.Count
                If Users(i).isOnline Then
                    If Users(i).Connection Is Nothing Then
                        Users(i).isOnline = False
```

```
            Else
               Users(i).Connection.GetWS.SendData "2" & Conn.User.UserName & "*"
& Mid(Data, 2)
               DoEvents
            End If
         End If
      Next
   Case 3
      Sleep 50
      Data = Mid(Data, 2)
      'extract data
      usr = BreakStr(Data, "*", 1)
      msg = BreakStr(Data, "*", 0)
      'search for user
      For i = 1 To Users.Count
         If Users(i).UserName = usr Then Exit For
      Next

      If i = Users.Count + 1 Then
         Conn.GetWS.SendData "0" 'user not found
         Exit Sub
      End If

      'if not connected user cannot be online
      If Users(i).Connection Is Nothing Then Users(i).isOnline = False

      If Not Users(i).isOnline Then
         Conn.GetWS.SendData "1" 'user is not online
         Exit Sub
      End If

      'everything is ok
      Conn.GetWS.SendData "2"
      DoEvents
      'send message
      Users(i).Connection.GetWS.SendData "3" & Conn.User.UserName & "*" & msg
      DoEvents
   Case 4
      Sleep 50
      Data = Mid(Data, 2)
      dat = Split(Data, "*")
      'search for user
      For i = 1 To Users.Count
         If Users(i).UserName = dat(0) Then Exit For
      Next

      If i = Users.Count + 1 Then
         Conn.GetWS.SendData "0" 'user not found
         Exit Sub
```

```
    End If

    'if not connected user cannot be online
    If Users(i).Connection Is Nothing Then Users(i).isOnline = False

    If Not Users(i).isOnline Then
       Conn.GetWS.SendData "1" 'user is not online
       Exit Sub
    End If

    dat(0) = Conn.User.UserName
    Data = Join(dat, "*")

    Users(i).Connection.SendingFile = Requested
    Conn.SendingFile = Requests
    Set Conn.FileUsr = Users(i)
    Set Users(i).Connection.FileUsr = Conn.User

    Users(i).Connection.GetWS.SendData "4" & Data
    pc = 0
Case 5
    DoEvents
    If Conn.SendingFile = No Then
       Conn.GetWS.SendData "9"
       Conn_Close Conn
       Exit Sub
    End If

    Conn.FileUsr.Connection.GetWS.SendData Data
Case 7
    Sleep 20
    If Conn.SendingFile = No Then
       Conn.GetWS.SendData "9"
       Conn_Close Conn
       Exit Sub
    End If

    Conn.FileUsr.Connection.GetWS.SendData "7"
    Sleep 20
    DoEvents
    Conn.FileUsr.Connection.SendingFile = No
    Set Conn.FileUsr.Connection.FileUsr = Nothing

    Conn.SendingFile = No
    Set Conn.FileUsr = Nothing
Case 8
    Sleep 20
    If Conn.SendingFile = No Then
       Conn.GetWS.SendData "9"
```

```
            Conn_Close Conn
            Exit Sub
        End If

        Conn.SendingFile = Recieving
        Conn.FileUsr.Connection.SendingFile = Sending
        Conn.FileUsr.Connection.GetWS.SendData "8"
    Case "-"
        Links(Mid(Data, 2) + 1).SendTo Conn
    Case "f"
        If Conn.SendingFile = No Then
            Conn.GetWS.SendData "9"
            Conn_Close Conn
            Exit Sub
        End If

        Conn.SendingFile = No
        Conn.FileUsr.Connection.SendingFile = No
        Conn.FileUsr.Connection.GetWS.SendData Data
        DoEvents
    End Select
Case notConnected
    Conn.GetWS.Close
End Select

Exit Sub
ErrH:
    log Err & Chr(9) & Err.Description
End Sub
```

JAVA Program Source-Code

```
/*
 * ChatClient.java
 *
 * Created on 07 Şubat 2004 Cumartesi, 14:51
 */

import java.awt.*;

public class ChatClient extends java.awt.Frame {

    public String Host;
    public ChatConn Conn;
    public List lst;

    /** Creates new form ChatClient */
    public ChatClient() {
        initComponents ();
        pack ();
            setTitle("Chat");
        setSize(512+120,252);
            lst=list1;
            txt=textArea1;
    }

    public void SetStatus(String txt)
    {
        label2.setText(txt);
    }

    public void fprog()
    {
    }

    public void inform()
    {
        switch(Conn.ConnectionState)
        {
            case 0:
                button1.enable();
                button2.disable();
                button3.disable();
                button5.disable();
                button6.disable();
                list2.disable();
                button7.enable();
                textField4.disable();
                break;
```

```
        case 1:
        case 2:
            button1.disable();
            button2.enable();
            button3.disable();
            button5.disable();
            button6.disable();
            button7.disable();
            list2.disable();
            textField4.disable();
            break;
         case 3:
            button1.disable();
            button2.enable();
            button3.enable();
            button5.enable();
            button6.enable();
            button7.disable();
            list2.enable();
            textField4.enable();
            break;
      }

      if(Conn.SendState>0 || (Conn.FileSend==1))
      {
                  textField4.disable();
                  button3.disable();
                  button5.disable();
                  button6.disable();
      }
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the FormEditor.
 */
private void initComponents()
{
          panel1 = new java.awt.Panel();
          button1 = new java.awt.Button();
          button2 = new java.awt.Button();
          button3 = new java.awt.Button();
          button5 = new java.awt.Button();
          button6 = new java.awt.Button();
          button7 = new java.awt.Button();
          panel2 = new java.awt.Panel();
          textArea1 = new java.awt.TextArea();
          textField4 = new java.awt.TextField();
```

```
label2 = new java.awt.Label();
lbl1 = new java.awt.Label();
panel3 = new java.awt.Panel();
list1 = new java.awt.List();
list2= new java.awt.List();
setLayout(null);
setBackground(new java.awt.Color (192, 204, 255));

panel1.setLayout(null);
panel1.setFont(new java.awt.Font ("Dialog", 0, 11));
panel1.setName("CommandPanel");
panel1.setBackground(new java.awt.Color (192, 204, 255));
panel1.setForeground(java.awt.Color.black);
      panel1.move(50,0);

button1.setFont(new java.awt.Font ("Dialog", 3, 11));
button1.setLabel("Connect");
button1.setActionCommand("connect");
button1.setName("cmdconnect");
button1.setBackground(new java.awt.Color (0x800a0e0));
button1.setForeground(java.awt.Color.white);
button1.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                connect_click(evt);
        }
}
);
panel1.add(button1);
button1.setBounds(0, 10, 80, 20);


button2.setFont(new java.awt.Font ("Dialog",3, 11));
button2.setLabel("Disconnect");
button2.setActionCommand("disconnect");
button2.setName("cmdDisconnect");
button2.setEnabled(false);
button2.setBackground(new java.awt.Color (0x800a0e0));
button2.setForeground(java.awt.Color.white);
button2.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                disconnect_click(evt);
        }
}
);
panel1.add(button2);
```

```
button2.setBounds(0, 40, 80, 20);


button3.setFont(new java.awt.Font ("Dialog", 3, 11));
button3.setLabel("Send File");
button3.setActionCommand("sendfile");
button3.setName("cmdSendFile");
button3.setEnabled(false);
button3.setBackground(new java.awt.Color (0x800a0e0));
button3.setForeground(java.awt.Color.white);
button3.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                sendfile_click(evt);
        }
}
);
panel1.add(button3);
button3.setBounds(0, 140, 80, 20);


button5.setFont(new java.awt.Font ("Dialog", 3, 11));
button5.setLabel("Send");
button5.setActionCommand("send");
button5.setName("cmdSend");
button5.setEnabled(false);
button5.setBackground(new java.awt.Color (0x800a0e0));
button5.setForeground(java.awt.Color.white);
button5.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                send_click(evt);
        }
}
);
panel1.add(button5);
button5.setBounds(0, 80, 80, 20);


button6.setFont(new java.awt.Font ("Dialog", 3, 11));
button6.setLabel("Send To");
button6.setActionCommand("sentto");
button6.setName("cmdSendTo");
button6.setEnabled(false);
button6.setBackground(new java.awt.Color (0x800a0e0));
button6.setForeground(java.awt.Color.white);
button6.addActionListener(new java.awt.event.ActionListener()
```

```
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                sentto_click(evt);
        }
}
);
panel1.add(button6);
button6.setBounds(0, 110, 80, 20);


button7.setFont(new java.awt.Font ("Dialog", 3, 11));
button7.setLabel("Host");
button7.setName("button14");
button7.setBackground(new java.awt.Color (0x800a0e0));
button7.setForeground(java.awt.Color.white);
button7.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                host_click(evt);
        }
}
);
panel1.add(button7);
button7.setBounds(0, 180, 80, 20);


add(panel1);
panel1.setBounds(3, 18, 90, 240);


panel2.setLayout(null);
panel2.setFont(new java.awt.Font ("Dialog", 0, 11));
panel2.setName("Texts");
panel2.setBackground(new java.awt.Color (192, 204, 255));
panel2.setForeground(java.awt.Color.black);

textArea1.setBackground(java.awt.Color.white);
textArea1.setEditable(false);
textArea1.setFont(new java.awt.Font ("Dialog", 0, 11));
textArea1.setForeground(java.awt.Color.black);
panel2.add(textArea1);
textArea1.setBounds(10+3, 18, 280, 170);


textField4.setBackground(java.awt.Color.white);
textField4.setName("textfield4");
textField4.setFont(new java.awt.Font ("Dialog", 0, 11));
```

```
textField4.setEnabled(false);
textField4.setForeground(java.awt.Color.black);
textField4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                send_click(evt);
        }
});
panel2.add(textField4);
textField4.setBounds(10+3, 180+18, 280, 20);


label2.setFont(new java.awt.Font ("Dialog", 3, 11));
label2.setName("label9");
label2.setBackground(new java.awt.Color (192, 204, 255));
label2.setForeground(java.awt.Color.black);
label2.setText("Not connected...");
panel2.add(label2);
label2.setBounds(10+3, 200+18, 280, 20);


add(panel2);
panel2.setBounds(80, 9, 310, 260);


panel3.setLayout(null);
panel3.setFont(new java.awt.Font ("Dialog", 0, 11));
panel3.setBackground(new java.awt.Color (192, 204, 255));
panel3.setForeground(java.awt.Color.black);

list1.setFont(new java.awt.Font ("Dialog", 0, 11));
list1.setBackground(java.awt.Color.white);
list1.setForeground(java.awt.Color.black);
list1.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                users_click(evt);
        }
}
);
    list1.setBounds(390, 28+20, 110, 210-20);


add(list1);

lbl1.setFont(new java.awt.Font ("Dialog", 3, 11));
lbl1.setName("label10");
lbl1.setBackground(new java.awt.Color (192, 204, 255));
lbl1.setForeground(java.awt.Color.black);
```

```
        lbl1.setText("No user");
        lbl1.setBounds(390, 28, 110, 16);
        add(lbl1);

        list2.setFont(new java.awt.Font ("Dialog", 0, 11));
        list2.setBackground(java.awt.Color.white);
        list2.setForeground(java.awt.Color.black);
        list2.disable();
        list2.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                Conn.senddata("-"+list2.getSelectedIndex());
            }
        }
        );
            list2.setBounds(390+116, 28, 110, 210);


        add(list2);

        addWindowListener(new java.awt.event.WindowAdapter()
        {
            public void windowClosing(java.awt.event.WindowEvent evt)
            {
                exitForm(evt);
            }
        }
        );
    }

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        System.exit (0);
    }//GEN-LAST:event_exitForm

    /**
    * @param args the command line arguments
    */
    public static void main (String args[]) {
        new ChatClient ().show ();
    }


    // Variables declaration - do not modify

  private void sendfile_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_sendfile_click
```

```
// Add your handling code here:
    if(!targetuser.equals(""))
    {
        FileDialog fd=new FileDialog(this,"Send file",FileDialog.LOAD);
        fd.show();
        Conn.sendFile(targetuser,fd.getDirectory() + fd.getFile());
    }
 }//GEN-LAST:event_sendfile_click

 private void sentto_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_sentto_click
// Add your handling code here:
    if(!targetuser.equals("") && !textField4.getText().equals(""))
    {
       Conn.SendTo(targetuser,textField4.getText());
       textField4.setText("");
    }
 }//GEN-LAST:event_sentto_click

  public String targetuser=new String();

 private void users_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_users_click
// Add your handling code here:
    targetuser=list1.getSelectedItem();
    lbl1.setText("selected: " + targetuser);
 }//GEN-LAST:event_users_click

 private void send_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_send_click
// Add your handling code here:
    if(textField4.getText().length()>0)
       Conn.Send(textField4.getText());
       textField4.setText("");

    this.sentto_click(evt);
 }//GEN-LAST:event_send_click

 public java.awt.TextArea txt;

 private void disconnect_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_disconnect_click
// Add your handling code here:
    Conn.Close();
 }//GEN-LAST:event_disconnect_click

 private void host_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_host_click
// Add your handling code here:
```

```
    ChatHost hostdiag=new ChatHost(this,true);
    hostdiag.Cparent=this;
    hostdiag.setHost(Host);
    hostdiag.show();
  }//GEN-LAST:event_host_click

  private void connect_click(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_connect_click
    ChatConnect conndiag=new ChatConnect(this,true);
    conndiag.Cparent=this;
    conndiag.show();
  }//GEN-LAST:event_connect_click

  private void links_click(java.awt.event.ActionEvent evt) {
  }

  // Variables declaration - do not modify
  public java.awt.List list2;
  public java.awt.Label lbl1;
  //GEN-BEGIN:variables
  private java.awt.Panel panel1;
  private java.awt.Button button1;
  private java.awt.Button button2;
  private java.awt.Button button3;
  private java.awt.Button button5;
  private java.awt.Button button6;
  private java.awt.Button button7;
  private java.awt.Panel panel2;
  private java.awt.TextArea textArea1;
  private java.awt.TextField textField4;
  private java.awt.Label label2;
  private java.awt.Panel panel3;
  private java.awt.List list1;
  // End of variables declaration//GEN-END:variables

}
```

CONCLUSION

The first chapter has information about A network connects computers by means of cabling systems, specialized software, and devices that manage data traffic. Computer networks fall into two main types: *client/server networks* and *peer-to-peer networks*. A client/server network uses one or more dedicated machines (the server) to share the files, printers, and applications. A peer-to-peer network allows any user to share files with any other user and doesn't require a central, dedicated server.

The second chapter I have talked about The evolution of Client-Server Computing has been driven by business needs, as well as the increasing costs for host (mainframe and midrange) machines and maintenance, the decreasing costs and increasing power of microcomputers and the increased reliability of LANs (Local Area Networks). In the past twenty years, there are dramatic improvements in the hardware and software technologies for microcomputers. Microcomputers become affordable for small businesses and organizations. And at the same time their performances are becoming more and more reliable.

The third chapter has explanation about TCP/IP that is not a single product. It is a catchall name for a family of protocols that use a similar behavior. Using the term TCP/IP usually refers to one or more protocols within the family, not just TCP and IP.
The OSI Reference Model is composed of seven layers. TCP/IP was designed with layers as well, although they do not correspond one-to-one with the OSI-RM layers. You can overlay the TCP/IP programs on this model to give you an idea of where all the TCP/IP layers reside.

Finally, chapter Four has described Client/Server Application and analyzed the application. I have developed a Java program to make connection between Client and Server for sharing any kind of information between a Client and a Server.

REFERENCES

[1] Michael Alexander, "*The Underground Guide to Computer Networks*", Fourth Edition, Addison-Wesley Press, November 1995.

[2] James Thomos, "*Data and Computer Communication*", Fourth Edition, North Holland publishing Company, August 1994.

[3] Tanebaum Andrew S., "*Computer Networks*", 1996

[4] *Cisco IOS Wide Area Networking Solutions*. Indianapolis: Cisco Press, 1999.

[5] David McDysan, Darren Spohn, "*ATM Theory and Applications*", McGraw-Hill, 1999.

[6] Larry L. Peterson, Bruce S. Davie, "*Computer Networks: A Systems Approach, Morgan Kaufmann Publishers*", Inc., 1996.

[7] Microsoft Windows 2000 network and Operating System Essentials, Workbook.

[8] Ross Ignall, Dranetz-BMI, "*Introduction to Network Communications*",1999.

[9] Rainer Händel, Manfred N. Huber, Stefan Schröder, "*ATM Networks: Concepts, Protocols, Applications*", Addison-Wesley, 1998.

[10] Douglas E. Comer, "*Computer Networks and Internets*" second edition 1999.

[11] Andrews S. Tanenbaum. "*Computer Networks*", third edition 1996.

[13] Stan Schatt, "*Understanding ATM*", McGraw-Hill, 1996.

[12] Handel Huber Schroder "*ATM Networks*", 1998.

[14] RAD Data Communications, "*ATM Tutorial*," 1994, http://www.rad.com/networks/1994/atm/tutorial.htm.

[15] William Stallings, "*ISDN and Broadband ISDN with Frame Relay and ATM*", Prentice-Hall, 1995.