11-22-1994

Fault tolerant Medical Network (MEDNET)

Hamid Ghassemi

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

FAULT TOLERANT MEDICAL NETWORK (MEDNET)

A thesis submitted in partial satisfaction of the

requirements for the degree of

MASTER OF SCIENCE

IN COMPUTER ENGINEERING

by

Hamid Ghassemi

1994

Thesis Committee Approval Sheet

To: Dean Gordon R. Hopkins
College of Engineering and Design

This thesis, written by HAMID GHASSEMI, and entitled, FAULT TOLERANT MEDICAL NETWORK (MEDNET) having been approved in respect to style and intellectual content, is referred to you for judgement.

We have read this thesis and recommend that it be approved.

Malcolm Heimer

Masoud Milani

Peter Romine

Subbarao V. Wunnava, Major Professor

Date of Defense: November 22, 1994

The thesis of Hamid Ghassemi is approved.

Dean Gordon R. Hopkins
College of Engineering and Design

Dr. Richard L. Campbell
Dean of Graduate Studies

Florida International University, 1994

Acknowledgments

I wish to thank my sister, Mehrangiz and my brother, Reza for their support.

I wish to thank the members of my committee: Dr. Heimer, Dr. Milani, and Dr. Romine for their help and guidance. A special thanks goes to my major professor, Dr. Subbarao Wunnava, for his wisdom, support and encouragement. I wish to thank all my friends, specially Parvaneh Mehdian, Minh Ly, Carlos Valdes, Felix Tong, Hiedi Savage, and Isidro Alvarez.

I also need to thank all the members of the ISDN lab, Kishore Gandham, Miguel Rosario, Daisy Barrera, Bruce Moreland, Dr. Irma Fernandez, Ramana Malneedi, Peter Hoo, whose help was invaluable to the completion of this project. I also wish to thank to everybody in FIU Electrical Engineering Department, specially Dr. Jim Story, Mike Urucinitz, Dr. Yen, Pat Brammer, Marbeth Cochran, and Laura Ruiz. I also like to thank Dr. Armando Barreto, Habibie and Kent Wilder, Steve Luis, Richard D'susa, Chris Edmon.

I wish to acknowledge Southern Bell and Northern Telecom for their technical and financial support required to establish the ISDN Laboratory.

ABSTRACT OF THE THESIS

FAULT TOLERANT MEDICAL NETWORK (MEDNET)

by

Hamid Ghassemi

Florida International University, 1994

Profssor Wunnava Subbarao, Major Professor

This investigation describes the development of a new fault tolerant Medical Network (MEDNET) model based on the existing Public Switch Telephone Network (PSTN), Integrated Services Digital Network (ISDN) and Internetworking (Internet). This research includes the original design, development and testing of the required hardware and software interfaces to provide a complete Medical Network model. MEDNET ties the Doctor, the Patient, the Hospital, the Medical Lab, and the Pharmacy for near real time and fault tolerant exchange of medical information. The MEDNET model includes the following modules: 1. Central Database Server, 2. Remote Client Access, and 3. Communication Interface. This work proves that medical images and data can be exchanged between healthcare providers which are not geographically adjacent, in a cost effective, timely, and secure manner.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ---------- 1

1.1 CURRENT SYSTEM LIMITATIONS ---------- 1

1.2 AN ALTERNATIVE SOLUTION ---------- 4

1.3 MEDNET BENEFITS ---------- 8

CHAPTER 2 MEDNET SYSTEM REQUIREMENTS ---------- 10

2.1 HARDWARE REQUIREMENTS ---------- 11

2.2 SOFTWARE REQUIREMENTS ---------- 12

2.3 SECURITY REQUIREMENTS ---------- 12

2.4 CONFIDENTIALITY REQUIREMENTS ---------- 13

2.5 EASINESS OF USE ---------- 13

2.6 COST REQUIREMENTS ---------- 14

CHAPTER 3 TECHNOLOGIES FOR IMPLEMENTING ---------- 20

SYSTEMS ---------- 20

3.1 NETWORK STRUCTURE ---------- 20

3.2 OPEN SYSTEMS INTERCONNECT(OSI) 7 LAYER MODEL ---------- 24

3.3 ISDN ARCHITECTURE ---------- 29

3.4 REFERENCE POINTS AND FUNCTIONAL GROUPINGS FOR ISDN ---------- 32

CUSTOMER PREMISES EQUIPMENT (CPE) ---------- 32

3.5 ISDN AND IT'S RELATIONSHIP WITH THE OSI MODEL ---------- 35

3.6 THE ISDN PHYSICAL LAYER ---------- 38

3.7 THE INTERNET NETWORK ---------- 41

CHAPTER 4 HARDWARE IMPLEMENTATION OF ---------- 42

THE SYSTEM ---------- 42

4.1 MEDNET ARCHITECTURE ----42

4.2 THE STANDARD TELEPHONE LINE ----45

4.3 THE INTERNET CONNECTION ----46

----47

FIGURE 4.4 MEDNET connection----47

4.4 THE ISDN CONNECTION ----47

4.5 MEDNET IMPLEMENTATION USING ISDN ----48

5.1) DSTART PROGRAM ----54

5.2 DOCTOR PROGRAM ----57
5.2.1 ADD A RECORD ----60
5.2.2 FIND A RECORD----61

5.3 DSERVER PROGRAM----61

CHAPTER 6 SYSTEM INTEGRATION ----63

6.1 HARDWARE AND SOFTWARE INTERFACES ----63

6.2 THE MEDNET FILE SERVER----67

6.3 MEDNET INTEGRATION ----69

6.4 MEDNET CASE STUDY----69

CHAPTER 7 CONCLUSIONS AND FUTURE WORK----71

7.1 CONCLUSIONS----71

7.2 FUTURE WORK----74

LIST OF REFERENCES----79

APPENDIX 'A'----84

CHAPTER 1 INTRODUCTION

Most medical establishments are independently computerized, and often these systems have their own local area network. Due to the cost and complexity of inter-connectivity, doctor's offices, hospitals, pharmacies and insurance companies have no true means of networking between each other. The result is a slow and costly process. The means of communication is most often via FAX and courier or mail services, which are timely and costly. On the other hand, communicating via the telephone has the disadvantage that it requires immediate attention. Many times these methods are inaccurate, inefficient and time consuming. FIGURE 1.1 shows the MEDNET architecture modeled at FIU's research laboratory.

1.1 CURRENT SYSTEM LIMITATIONS

As mentioned earlier, most doctors, hospitals, pharmacies, and laboratories are not internetworked.

Information transfer is very slow, costly and inefficient. For example, if a doctor needs to ask a question or opinion of another doctor, both doctors must be free in order for this communication to take place. When the doctor's office wants to send a patient's file to another doctor's office, usually the file has to be printed from the computer then to be mailed or faxed. This is very time consuming, since the speed of the process depends on how busy and how fast the person can process the request. Human intervention not only slows down the process, but can result in errors.

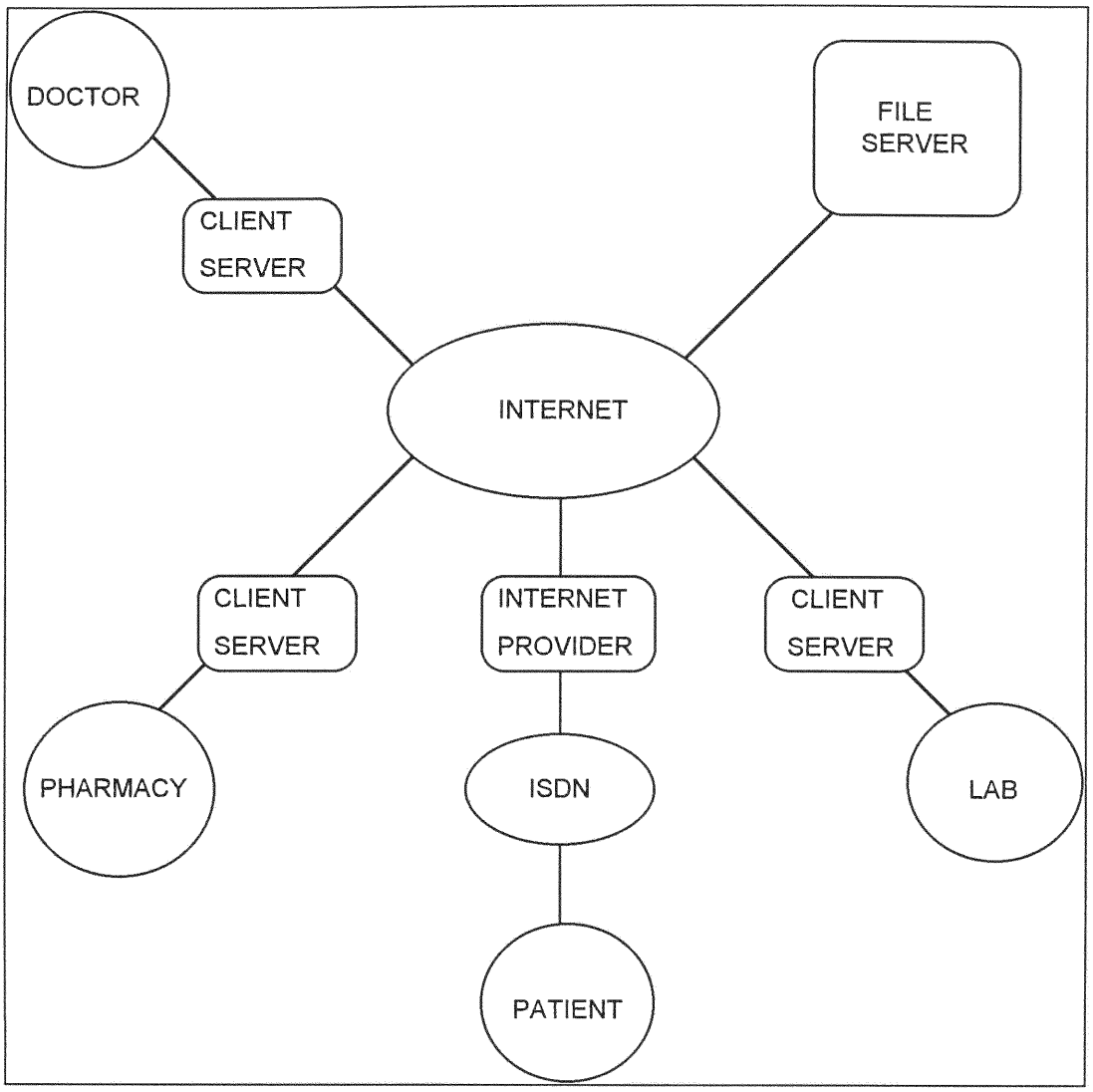


FIGURE 1.1 MEDNET Architecture

Other areas of application are the pharmacies and medical laboratories. In the particular case of the pharmacies, most pharmacies are not interconnected and do not have a centralized patient data base. In general, pharmacies are not interconnected to the doctor's offices.

If a patient goes for a medication refill to any pharmacy other than the one that he/she originally goes to, the pharmacy will have to make a phone call to the store that has the copy of the prescription, and get more information about the validity of the prescription. This involves extra time and cost (the cost of the employee time spent in verifying the prescription). Another problem is the possibility of forging and/or altering the doctor's prescriptions. When the doctor hands in the prescriptions to the patient, the patient has in theory, the opportunity to alter the prescriptions. Because of that possibility, in the case that the drug is narcotic, the pharmacist will have to make a telephone call to the doctor to verify the validity and dosage of the prescription. The best case scenario is that the patient will have to take the prescription to the pharmacy in person, stand in the line for several minutes, wait for some more time until the medicine is ready, and wait in another line to pick up the medicine. This process is lengthy and inefficient.

1.2 AN ALTERNATIVE SOLUTION

Using a conventional dial-up method to connect all the doctors to all the pharmacies can partially solve the

interconnectivity problem. This solution would require large number of telephone lines for pharmacies and extra telephone lines for doctor's offices. The cost involved in providing the necessary extra telephones lines makes this an impractical solution.

ISDN can offer the lowest cost and most efficient solution to the interconnectivity problem. One complete ISDN (Integrated Services Digital Network) BRI line will provide voice and data capabilities on two B-channels and one D channel that can be used to connect to one or more computers. B-channels work at 64K bps each, and D-channel works at 16k bps to carry signaling information for the associated B channels and access the packet network. FIGURE 1.1 shows the Medical Network (MEDNET) architecture.

In the case of medical laboratories, this problem caused by lack of interconnectivity is even worse. The patient usually has no access to laboratory records. This problem is aggravated if the medical lab is out of town, or out of state. For fast and efficient medical service, the doctor should have immediate access to the patient's medical record and medical laboratory record. The current method is that the doctor's office would have to call the medical lab and provide all the information to the person in charge at the lab. The best case scenario is that the lab technician would search for the patient's lab result in the computer, then print the result, and read it over the telephone or fax it

to the doctor's office. This is an inefficient and time consuming process, and usually the patient has to make another visit to the doctors office to discuss the lab results once the doctor has received this information.

The proposed ISDN based MEDNET will allow the doctor's office to call the laboratory simply by clicking on a icon on the screen. The doctor's office hardware requirement is a subscription to Basic Rate Interface (BRI) service ISDN, and a computer. Currently, virtually all doctor's offices have computers and a piece of communication software. The ISDN line can provide two voice channels (two B-channels) and one data (Packet D-channel). On the laboratory site, one ISDN BRI would be enough. Two B-channels can provide access at 64 KBPS, and the D-channel with it's 16 KBPS, will support signaling and provide access to the packet network. One BRI in packet mode, so it could allow access up to 128 users at a time with adaptive data rate. Adaptive data rate means the connection speed for each user changes as the number of users that are connected to the system changes. Adaptive data rate means the connection rate per user drops when the number of connections increases, and increases when the number of connections decreases.

Integrated Services Digital Network, ISDN, provides a low cost link to effectively integrate MEDNET Doctors, Medical Laboratories, Pharmacies, Hospitals and Insurance companies. This network will be based on ISDN communication

technology. FIGURE 1.2 shows the typical ISDN BRI architecture.

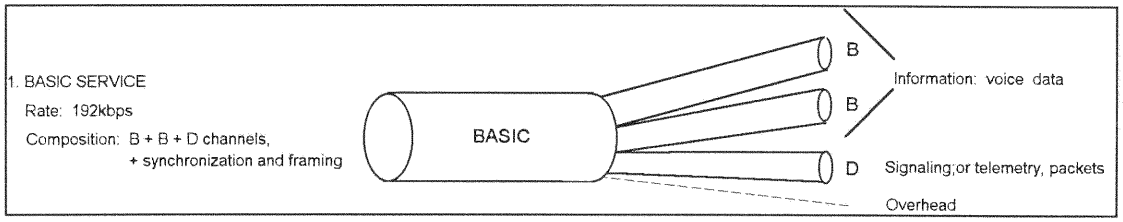


Figure 1.2 Typical ISDN BRI architecture (Courtesy STAL92)

FIGURE 1.3 shows the D channel utilized at the doctor's office to communicate with the other doctors for E-mail purposes, and also with other medical centers like medical labs and pharmacies. Medical centers which serve the database, would use one complete BRI (2B+D) in Packet mode to accommodate a large number of users connected at the same time. In this thesis we develop, implement an test the operating model of MEDNET.

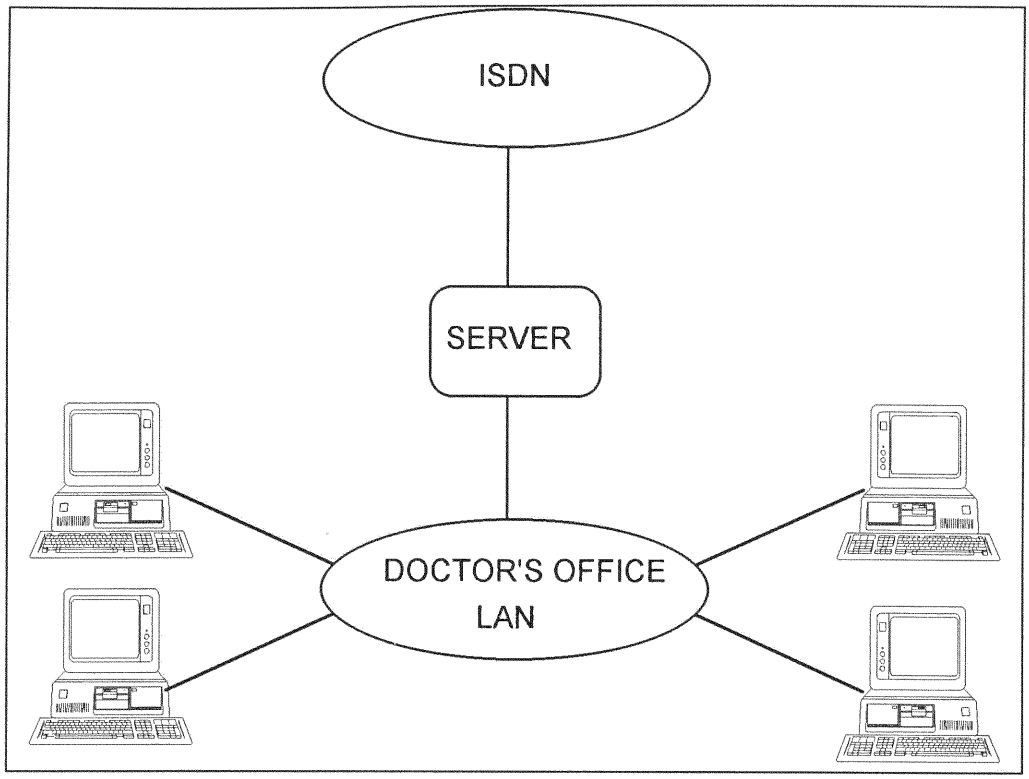


FIGURE 1.3 Doctor's office communication architecture

1.3 MEDNET BENEFITS

MEDNET will network doctors, pharmacies, hospitals, insurance companies, laboratories, in short any healthcare provider. MEDNET provides a higher security for doctors issuing prescriptions drugs. Through Internetworking of the doctor and the pharmacy, prescription can not be forged or modified by patients. The doctor would be able to send

medical information and prescriptions immediately in a secure fashion. Furthermore, the patient would not have to make two trips to the pharmacy, they would only have to go to pick up the prescription.

CHAPTER 2 MEDNET SYSTEM REQUIREMENTS

In this chapter the system requirements regarding the following categories are discussed.

1. Hardware: MEDNET hardware requirement can be met with developing communication technologies like Internet, ISDN, and other fast growing communication technologies.

2. Software: MEDNET software requirements can be provided with an appropriate operating system, data base access, graphics software and communications software.

3. Security: MEDNET security requirements with respect to access to patient information and maintaining the integrity of the patient's data are essential.

4. Confidentiality: MEDNET confidentiality requirement of the patient information for privacy purposes has to be implemented. Currently there is no existing off-the-shelf software to meet these requirements.

5. Ease of use: The MEDNET patient/doctor interaction must be convenient, accurate, and response time must be quick. Furthermore, system commands must be user transparent.

6. Cost saving: The MEDNET must be a cost savings system for the doctor, the patient, the insurance companies, the hospital, the medical labs, and the pharmacies.

7. Expandability: The complete MEDNET system must be expandable as a medical organization.

2.1 HARDWARE REQUIREMENTS

Although MEDNET is network independent, here at FIU's ISDN Lab we have implemented MEDNET using the ISDN and Internet network communication technologies. One reason for this decision is the availability of these communication technologies right here at our Lab. This enables us to realistically test the proposed Medical network. Other network technologies such as Asynchronous Transmission Mode (ATM) and frame relay may also be used as these communication technologies become available.

The concept of the MEDNET system is portable to different network platforms. The file server computer can be any system that can run the UNIX operating system. We implemented the file server with a Sun Sparc 10 with 64 Megabytes Random Access Memory (RAM) and one Gigabyte of hard drive. NCD X-terminal and PC-compatible computers with

X-terminal software running under windows, interface to MEDNET.

2.2 SOFTWARE REQUIREMENTS

Any multitasking and multi-user operating system such as UNIX, or windows NT is well suited for MEDNET model implementation. We choose to use UNIX operating system for server computer under NFS networking. We developed the concept and completed the operating software for the dynamic data base implantation with GCC, the C++ compiler. For fast data access, the binary-tree structure with automatic B-tree balancing algorithm is used. On the remote site we used the PC computer with X-terminal graphics software under the Windows operating system, NCD X-terminal and Sun3/80.

2.3 SECURITY REQUIREMENTS

Due to the nature of the sensitivity of the patient's information, security is an important part of MEDNET. There are several levels of security checks. The first security check is the access password to the UNIX server system. The

second security check is the MEDNET password check. The third security check is the patient's authorization check.

Doctors don't have access to any patient file unless he or she has the patient's permission code, and no one has authorization to alter records other than the authorized medical staff.

2.4 CONFIDENTIALITY REQUIREMENTS

Due to the patient's requirements for privacy of information, extreme care is taken to guarantee data confidentiality. This is done by multilevel checks and by authorization codes. These codes are made dynamic so that breaking the code becomes almost impossible.

2.5 EASINESS OF USE

The MEDNET is very easy to use. At the server side the system is fully automatic. In fact the system stays up and running at all time. At the client or user site, the person logs-into the server by entering the user name and password. The system will invoke the client program from the user's login file. Once the client program is invoked, the user will be provided with a user friendly menu to make the

proper selections. This is essential since the patients and medical personal need not to be computer professionals, and therefore the MEDNET system should be a simple turn key system.

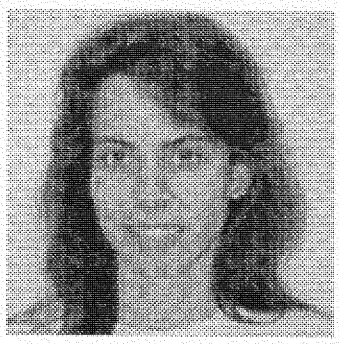
2.6 COST REQUIREMENTS

Cost is usually one of the most important factors in promoting and accepting any new system. One interesting feature of MEDNET is that it's cost saving for patients and doctors. For example, when a person goes to visit a doctor for the first time, the diagnosis is not finished in that first visit. Normally the doctor asks the patient to come back again for another visit, after the doctor has had the chance to request and receive the patient's previous records. This information has to come from the medical labs or other doctors, who have examined the patient. In most cases this second visit could be eliminated and saves the patient and his or her insurance the cost of a second visit. MEDNET saves cost to medical institutions by reducing the redundant paper work. For example, if the patient information is in the doctor's computer, the system can easily manipulate the data to accommodate insurance forms, lab work forms, or hospital admission forms. This process

of automation and standardization is a key characteristic of the MEDNET.

The following FIGURES 2.2, 2.3, 2.4, 2.5, and 2.6 show some of the records which are standardized in MEDNET.

patient information



Record number:

First Name: Heidi
Last Name: Smith

social security number:012-34-5678
insurance company: Dimah Insurance Co.
policy number: 7583-98234

telephone (day time) : 555-1111
(night time): 123-6567

emergency contact name: 454-3344
emergency contact tel.: 345-6789

date of birth: 9/4/1968

address: 1234 SW 8 Ave
Miami, Fl, 33175

Additional information:

Physician information

doctor's

Doctor's Name:

license number:

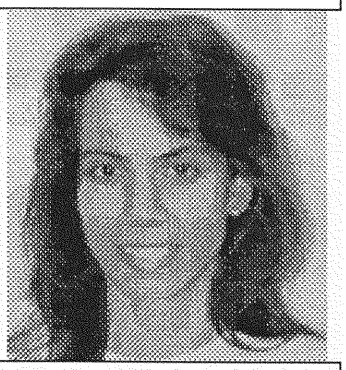
phone (day):
(night):

address:

hospital affiliations:

insurance affiliations:

Laboratory Information



DISPLAY: icrl	ssn# 012-34-5678
first name: Heidi	last name: Smith
address:	12345 SW 8 Avenue
	Miami, FL 33174
phone:	555-

test:

test result:

Figure 2.4 Laboratory Form

Pharmacy Information

Patient's Name: Heidi Smith
Doctor's Name: Dr. Subbarao
Patient's ssn# 012-34-5678
Date: 11/22/94
Medication: Aspirin

Figure 2.5 Pharmacy Form

CHAPTER 3 TECHNOLOGIES FOR IMPLEMENTING SYSTEMS

Networking is the backbone of the MEDNET development. Networking is a resource sharing system. Two types of communication technology are being used to implement the MEDNET system. They are the Integrated Services Digital Network (ISDN) and the Internet Network.

3.1 NETWORK STRUCTURE

There are many advantages of using a network architecture. The first advantage of using a network is that it makes all the programs, data and equipment on the network available to anyone regardless of physical location of resource or user. The second advantage is that it provides higher reliability. For example all critical files when updated, could be stored on redundant databases for fault tolerance purposes.

A Network Structure is defined as the means of communication between two computers. In today's technology, there are various topologies for information exchange. The

two types of communication structure in computer networks are as follows:

1. Point-to-point channels.
2. Broadcast channels.

Point-to-point channels structure is used for transmitting messages, in packets, from one Interface Message Processor (IMP) to another. An important design issue for a point-to-point subnet is interconnection topology. FIGURE 3.1 shows the star, ring, tree, complete, intersecting rings, and irregular topologies.

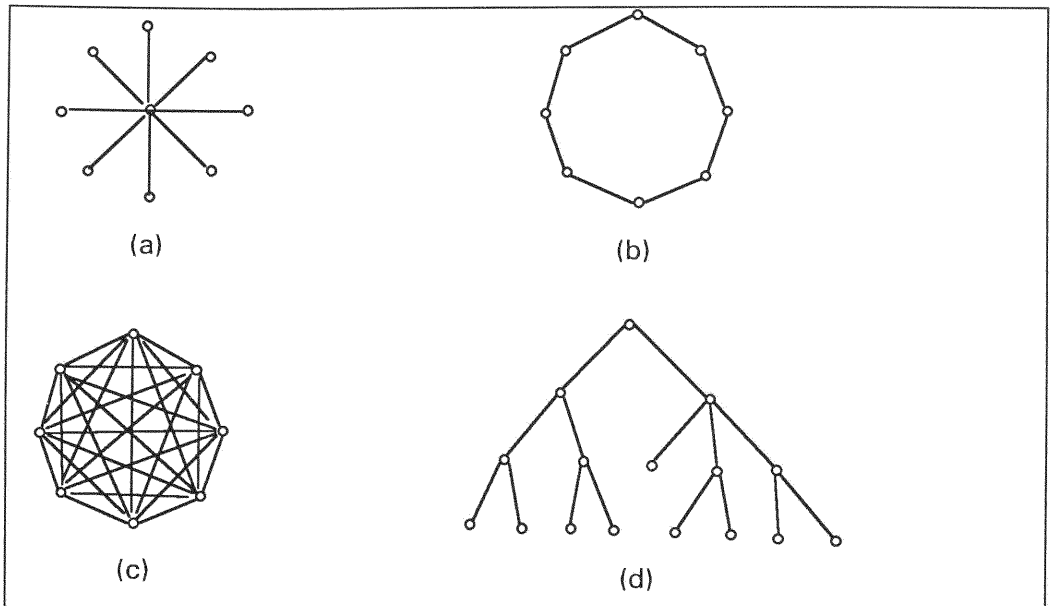


Fig. 3.1 Some possible topologies for a point-to-point subnet. (a) Star. (b) (c) Complete. (d) Tree. (courtesy [andr88])

A Broadcast channel structure consists of a single communication channel that is shared by all the computers on the network. Messages, in packets, are transmitted by one computers and received by all other computers. The packets contain the address of the intended computer. Once a packet is received, each computer validates the address field. If the packet is intended for another system, it is discarded.

Broadcasting allows the possibility of transmitting packets to all destinations by using a special code in the address field. Upon receiving a packet with the special code, it is accepted and processed by every system on the network. This feature is called multicasting. FIGURE 3.2 shows the communication network using broadcasting through bus, satellite or radio and ring.

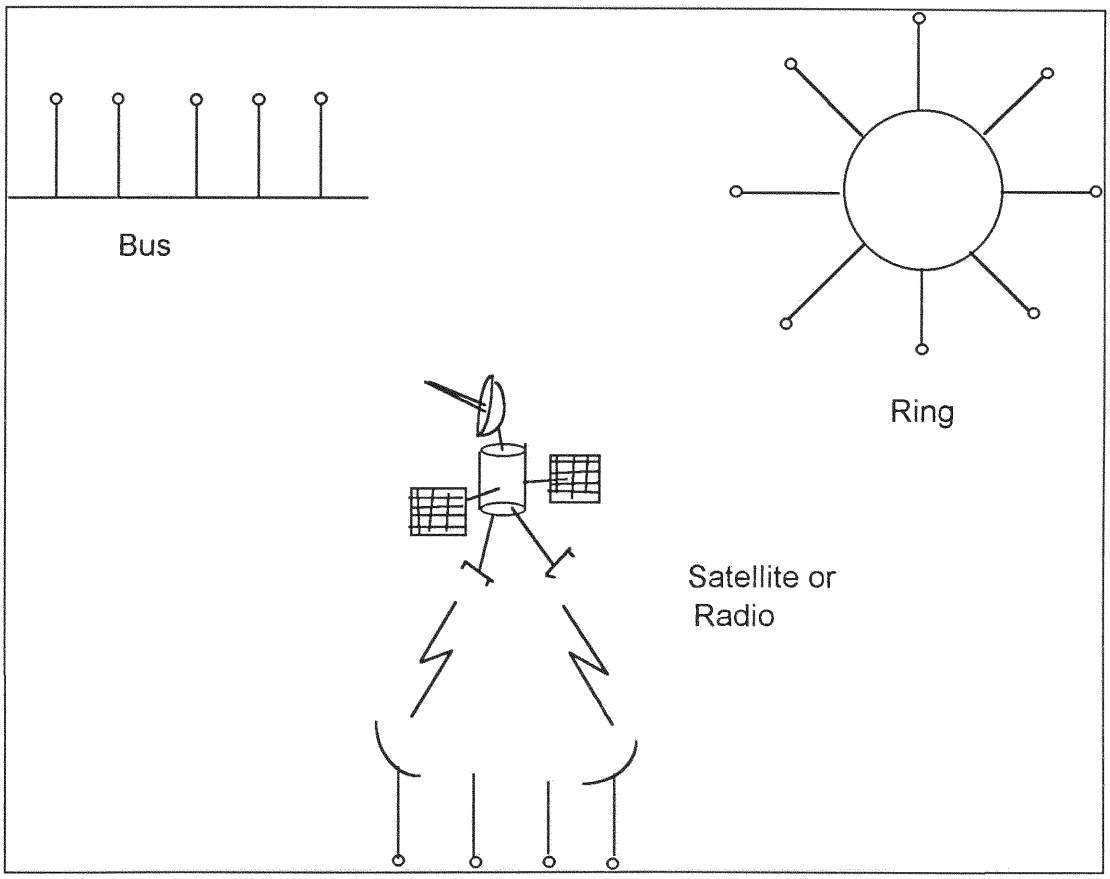


FIGURE 3.2

3.2 OPEN SYSTEMS INTERCONNECT(OSI) 7 LAYER MODEL

In 1978 the International Organization for Standardization (ISO) proposed establishing a framework for developing standards for future interconnection of heterogeneous Open Systems. "Open systems" refers to computers or systems which conform to the reference models and the associated standards to inter-connect [ISO 7498]. FIGURE 3.3 represents the Open Systems Interconnect Model.

The ISO reference model divides the communications functions hierarchically into seven layers. Each layer provides service to the next higher layer by using the services that are provided to them by the previous layer.

Layer seven, the Application layer, serves an application by providing communication support. The application does not reside in layer seven even though the communication is attached to layer seven. In reality, one can divide the application into a communication part and a non-communication part. The communication can be defined as an application entity. The Application Layer derives from the Presentation layer (layer six). An example of the communication component in the MEDNET system is that doctors can use an electronic mail (e-mail) to communicate with the Hospital or Pharmacy regarding the patients. As for the non-communication part, an individual user can access general software for office use.

Layer six, the Presentation layer, is used to deal with information exchanged between two system so that the original data can be preserved even though the individual systems are different. When the two systems try to establish a connection, they will negotiate on a particular data transfer format, such as encryption technique, compression technique, or reformatting. Once a particular format is agreed upon, both systems will be able to perform the operation and reconstruct the data back to it original format.

Layer five, the Session layer, is used to organize and to synchronize communication between two applications in two different systems on a network. The Session layer manages data exchange. In other words, it provides different types of communication such as full duplex or half duplex. The Session layer provides each user with a token which represents authorization for a user to transmit the data. Also, it provides the session service for users to exchange limited amount of control information while not within the activity.

Layer four, the Transport layer provides reliable end-to-end service between users. The Transport layer can be classified as follows:

1. Connection Oriented Transport Service(COTS)
2. Connectionless Transport Service(CLTS)

In Connection Oriented mode, the two transport service users are provided with full duplex transmission. On the other hand, Connectionless Transport Service uses package switching.

Layer three, the Network layer, is a service that is provided to the upper layer to exchange information without the concern about physical and data transmission and switching technology. The network layer also provides a global addressing scheme so end systems in different subnetworks can be addressed.

Layer two, the Data Link layer, is responsible for reliable and error free data transfer on the data link. It provides functions such as establishing the link, error detection, error recovery, and flow control.

Layer one, the Physical layer deals with mechanical and electrical characteristics of the physical link such as the connecting media and voltage level of the signal. This layer is concerned with transmission of raw data and bit streams.

The ISDN structure complies with layers 1 through 3 of the OSI architecture for Packet Switched Data (PSD) connections. Other 4 layers are user controlled for ISDN transmission.

Layer 1 of ISDN is defined for Circuit Switched Data (CSD) or Circuit Switched Voice (CSV) connections. Therefore, ISDN is not concerned with layers 4 through 7 of the OSI model, which are essentially user defined.

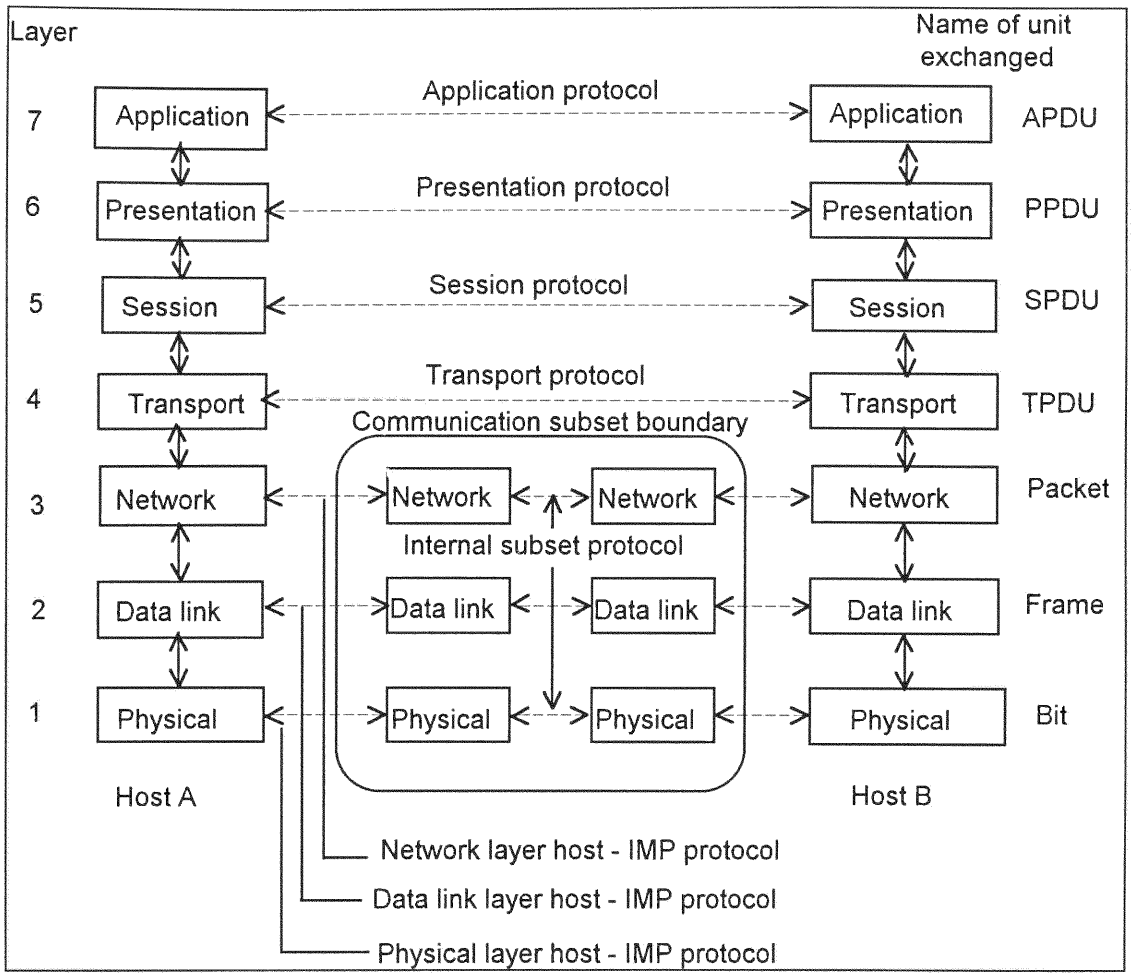


FIGURE 3.3 Open Systems Interconnect Model

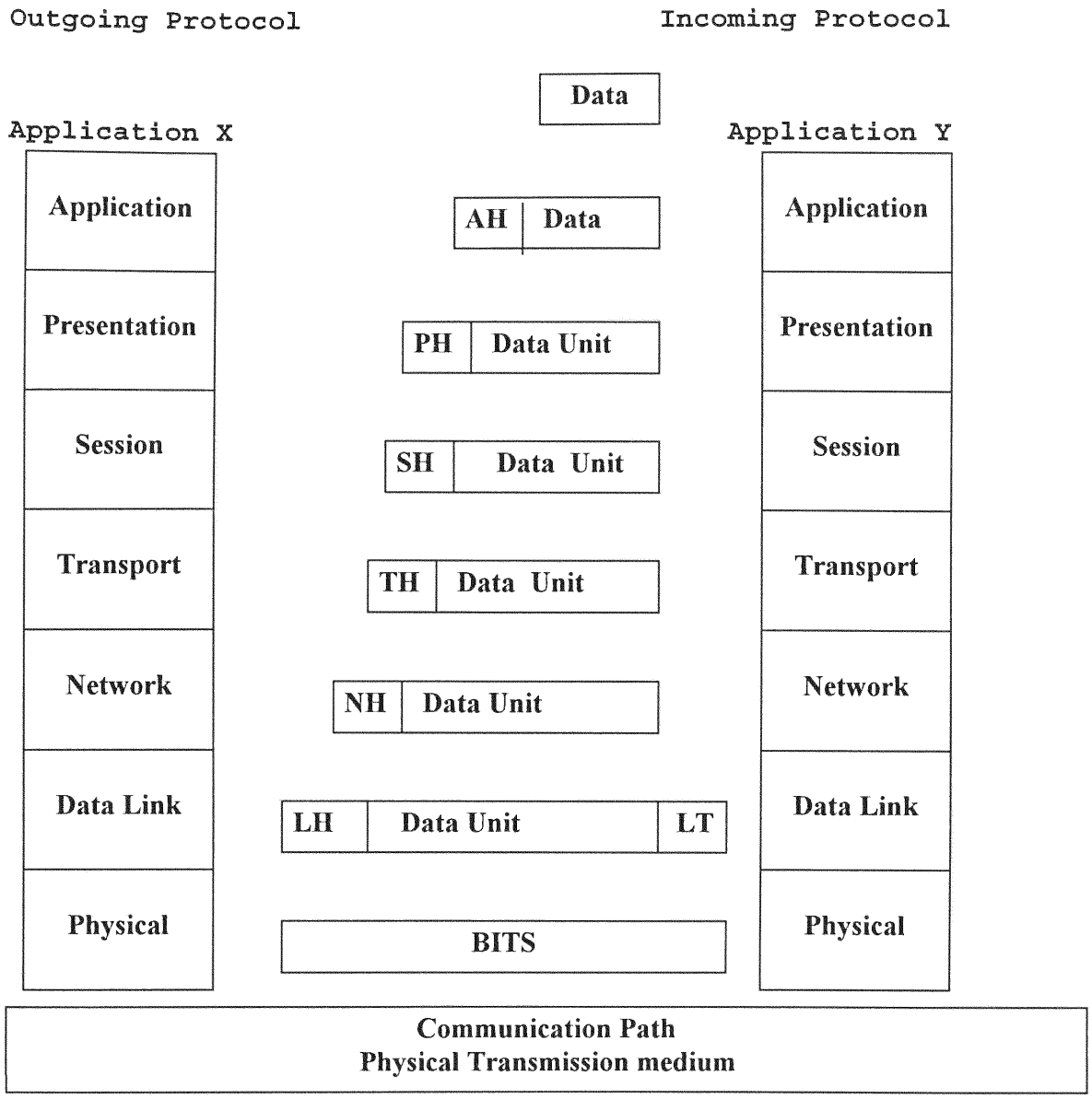


FIGURE 3.4 OSI Operation (Courtesy [Folt83])

Figure 3.4 illustrates the principles of OSI. Basically, if an application X (AP'X') has a message (AP data) for an application Y (AP'Y'), the data is transferred first to the Application layer 7 which appends a header (AH) to the data and transfers the two (AP data & AH) to the Presentation layer 6.

The presentation layer appends it's own header (PH) and transfers down to the Transport layer 5. The process continues through to the Data layer 2 where the frames include a header (F,A,C) and a trailer (FCS,F). The Physical layer 1 then transfers the frames across as a bit stream to the physical medium to layer 1 of user Y. At this point, the reverse process occurs, as each layer strips off the header to pass the information up to the next layer, until it reaches Application Y (AP'Y').

3.3 ISDN ARCHITECTURE

The transmission structure for ISDN is built upon communication channels. The types of channels, as defined by CCITT, are:

1. Binary (B) channels: Supports full duplex transfers up to 64 KBPS, the standard user channel rate. This rate was chosen because it is the most effective for digitized voice. B channels can be used for digitized voice or data.

The B channel can be used to make three kind of connections: Circuit-switched Voice (CSV), Circuit-switched Data (CSD), and Packet-switched Data (PSD). Circuit-Switching is defined [Stal92] as: "A method of communicating in which a dedicated communications path is established between two devices through one or more intermediate switching nodes...digital data is sent as a continuous stream of bits. Data rate is guaranteed, and delay is essentially limited to propagation time". On the other hand, Packet-Switching is defined [Stal92] as: "A method of transmitting messages through a communications network, in which long messages are subdivided into short packets. Each packet is passed from source to destination through intermediate nodes. At each node, the entire message is received, stored briefly, and then passed on to the next node." The user is connected to a packet network using X.25 protocol. Traditionally, billing for CSV and CSD have been on basis of connection time, while PSD is billed on the basis of transaction data rates.

2. Delta (D) channels: A 16 KBPS which is used for two purposes. First, the D channel carries the common-channel signaling information required to establish the circuit-switched calls on the associated B channels. Second, the D channel may also provide an access to the packet-switched network, with a maximum throughput of 9.6 KBPS, the remaining bandwidth is taken up by the control signaling.

The D channel can also be split among up to 6 devices accessing the packet-switched network. In this configuration, the bandwidth is dynamically allocated amongst the active devices.

3. H channel: Used for higher bit rate speeds, H0 supports 384 KBPS, H11 supports 1536 KBPS, H12 supports 1920 KBPS.

The number of channels carried to the subscriber will depend upon to the type of service the user subscribes to. At this time, there are two possible types of service to which a user may subscribe:

1. Basic Rate Interface (BRI) is the most cost effective and widely deployed with two B channels, and a D channel operating simultaneously the existing telephone line. The total bit rate, including the overhead due to framing and synchronization, is 192 KBPS.

2. Primary Rate Interface (PRI) is appropriate for larger bandwidth applications. with 23-B channels and an associated D channel operating at 64 KBPS. The total bit rate including framing and synchronization is 1.544 MBPS. This bit rate definition for PRI is valid in the US, Japan and Canada. In Europe PRI is defined as 30-B channels and 1-D channel, for a total throughput of 2.048 MBPS. For still larger bandwidth applications in excess of 100 MBPS, Broadband ISDN (B-ISDN) can be used. FIGURE 3.x represents the ISDN Architecture.

3.4 REFERENCE POINTS AND FUNCTIONAL GROUPINGS FOR ISDN

CUSTOMER PREMISES EQUIPMENT (CPE)

FIGURE 3.6 is a generic representation of ISDN reference points and functional groupings for CPE required for ISDN user access. The ISDN Reference interface points are as labeled R, S, T, and U. The Functional groupings are Network Termination 1 (NT1), Network Termination 2 (NT2), Terminal Equipment 1 (TE1), and Terminal Equipment 2 (TE2).

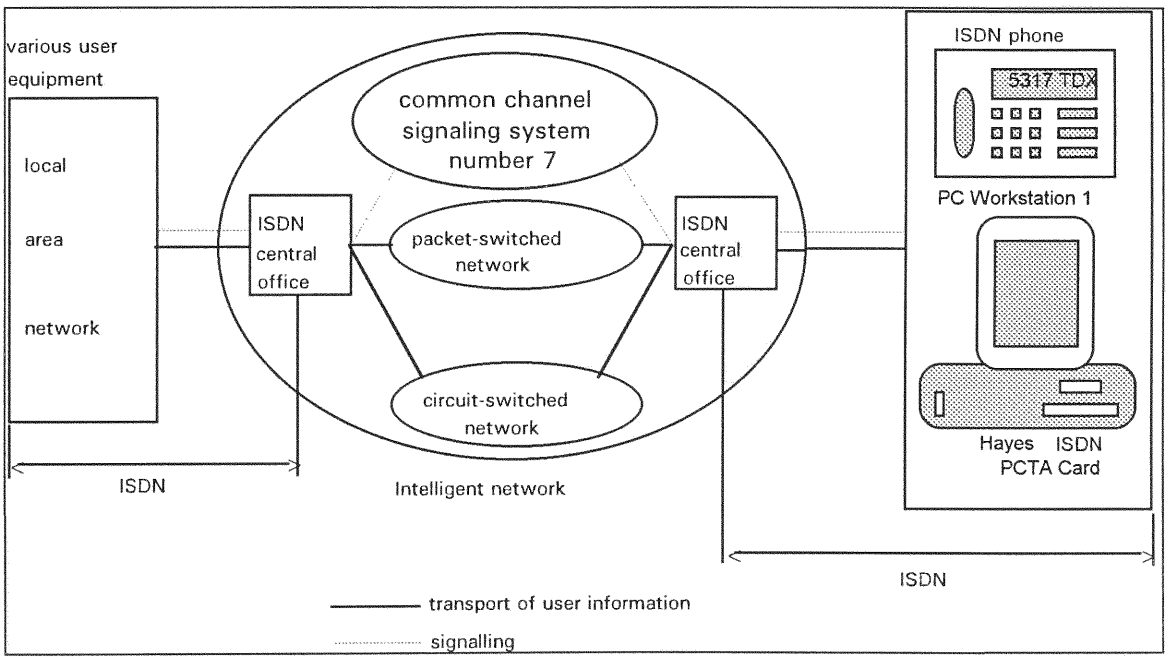
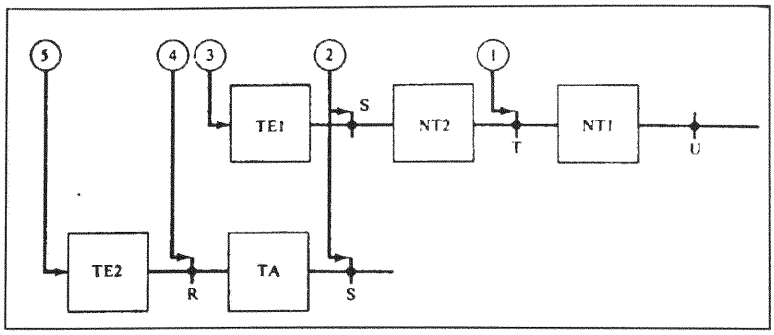


FIGURE 3.5 ISDN Architecture (Courtesy [Fern94])



TE1 = Terminal Equipment type 1 TE2 = Terminal Equipment type 2
NT1 = Network terminator 1 NT2 = Network terminator 2
TA = Terminal Adapter
Reference Interface Points: R (Rate), S (System), T (Terminal), U (User)

FIGURE 3.6 ISDN Reference Points and Functional Groupings (Courtesy [Stal92])

The NT1 provides a termination to the two wire transmission line from the local network. The NT1 uses synchronous time-division multiplexing to multiplex/demultiplex the bit streams of multiple channels together at the physical layer 1. The NT2 performs switching and concentration functions, supporting layers 1 through 3, for example for a digital PBX.

The User (U) reference interface is the full duplex link at the subscriber loop. The Terminal (T) reference point sets the demarcation between the ISDN and the user's equipment. The System (S) reference point corresponds to the user ISDN terminal equipment. The Rate (R) reference point is an interface for non-ISDN compatible equipment. A Terminal Adapter (TA) is required to connect non-ISDN compatible equipment (TE2), such as an analog phone or a

personal computer to the ISDN. TE1 equipment refers to ISDN compatible equipment such as an ISDN telephone.

In a typical BRI application the NT2 function is not required, and therefore the S and T interfaces appear together as the S/T Bus. The CCITT reference configuration for BRI is depicted in FIGURE 3.7. As we can see, the Network Termination 1 (NT1) provides the termination to the two-wire transmission line from the local network and connects to a 4 wire S/T bus. This bus can be configured in point-to-point or multipoint-to-point mode. In point-to-point, one TE is connected at the end of the S/T bus, which can be no longer than 1 km. In the multipoint-to-point configuration, up to eight terminals can be connected in parallel to the S/T bus, but the bus can't be longer than 200 meters. The rate at the S/T bus is the sum of 2 B + 1 D + framing and synchronization for a total of 192 KBPS.

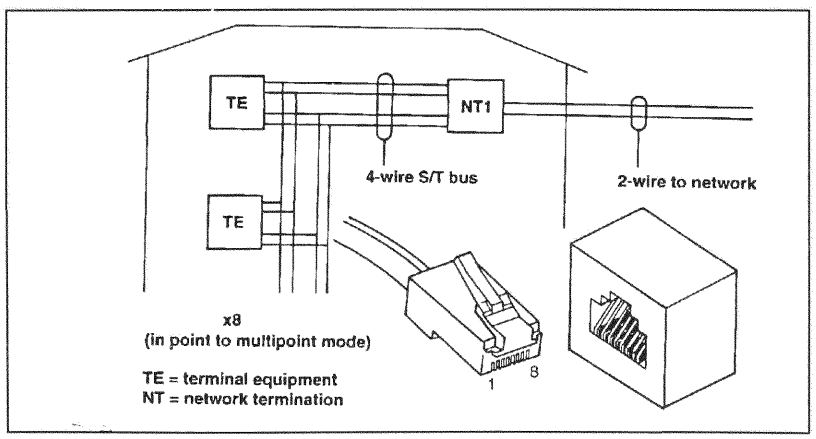


FIGURE 3.7 BRI Configuration (Courtesy [Grif90])

3.5 ISDN AND IT'S RELATIONSHIP WITH THE OSI MODEL

The ISDN structure defines layers 1 through 3 of the OSI architecture for PSD connections, and layer 1 for CSD or CSV connections. Therefore, ISDN is not concerned with layers 4 through 7 of the OSI model, which are essentially user defined.

For circuit switched connections, the establishment of the circuit is not done over the B channel, but over the D channel via Common Channel Signaling System No.7 (SS7). Once the connection is established, the network appears at layer 1. FIGURE 3.8 represents the network configuration and protocols for circuit switched calls. Table 3.1 is the key for Figures 3.8, 3.9, and 3.10.

SS7 is a four-layer protocol which controls both telephone voice and digital data connections. The SS7 control signaling is implemented using a packet-switching technology. This means that control messages to implement call management (setup, maintenance, termination) and network management functions are routed as packets through the network. In essence, a packet-switched network is overlaid on a circuit-switched network in order to operate and control the circuit-switched network [Stal92].

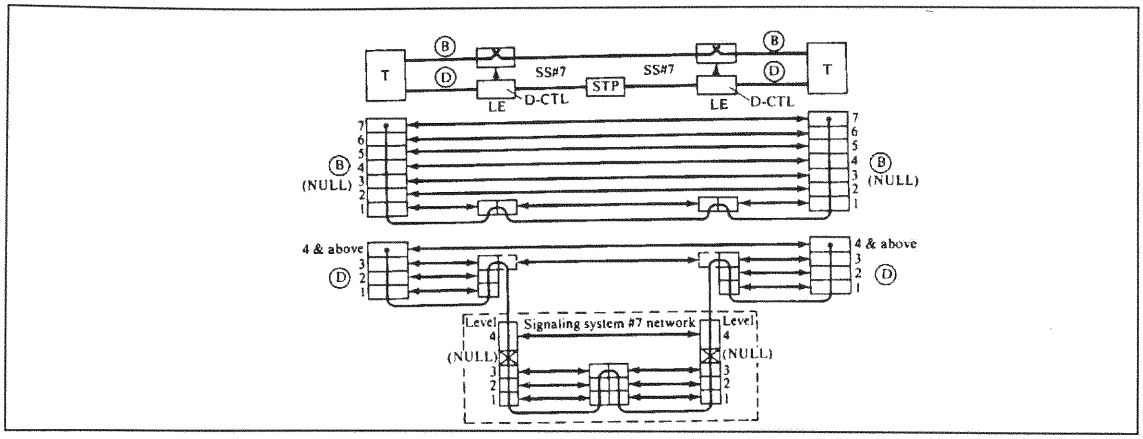


FIGURE 3.8 Network Configuration and Protocols for Circuit Switching (Courtesy [Stal92])

B = An ISDN B Channel	D = An ISDN D Channel
T = Terminal	D-CTL = D channel controller
SS7 = CCITT Signaling System 7	STP = Signaling Transfer Point
(Null) = Channel not present	7,6,5,4,3,2,1 = Layers in ISO model
LEVEL = Levels in SS7	LE = Local exchange
TE = Transit exchange	PSF = Packet-switching facility
Horizontal line = Peer-to-peer protocol	Vertical line = Layer-to-layer data flow

Table 3.1 Key to Figure 3.8, 3.9, 3.10

For Packet switched connections, there exist two possibilities. First is where a B-channel is used to access a packet switching network. For the region served by Southern Bell, the packet-switching capability is not integrated into ISDN but provided by a packet-switched public data network (PSPDN). In this case, the user

requests a circuit-switched connection on a B channel to the packet network via the D channel. The connection is then set up using SS7 and the user is notified through the D channel. The user sets up a virtual circuit using X.25 protocol for call establishment and LAPB (Link Access Protocol - B channel) to connect to the packet network. FIGURE 3.9 represents the network configuration and protocols for packet switched calls using the B channel.

The second possibility is where a D-channel is used to access a packet switching network. Here the user sets up the virtual circuit using X.25 for call establishment and LAPD (Link Access Protocol - D channel) to connect to the packet network. FIGURE 3.10 represents the network configuration and protocols for packet switched calls using the D channel. LAPD is based on the HDLC (High Level Data Link Control) protocol, except it incorporates a two address field to differentiate between a call control procedure or a packet communication, the two possible types of traffic on the D channel.

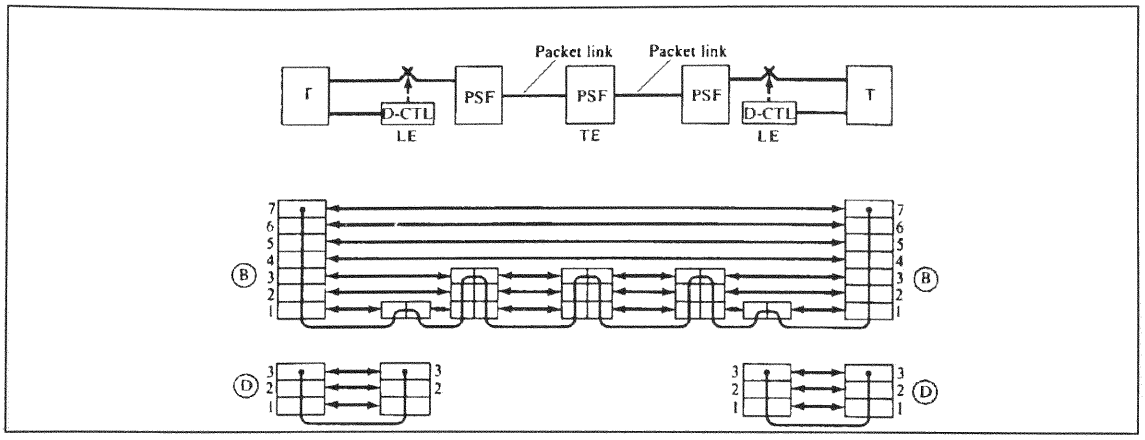


FIGURE 3.9 Network Configuration and Protocols for Packet Switching Using B Channel (Courtesy: [Stal92])

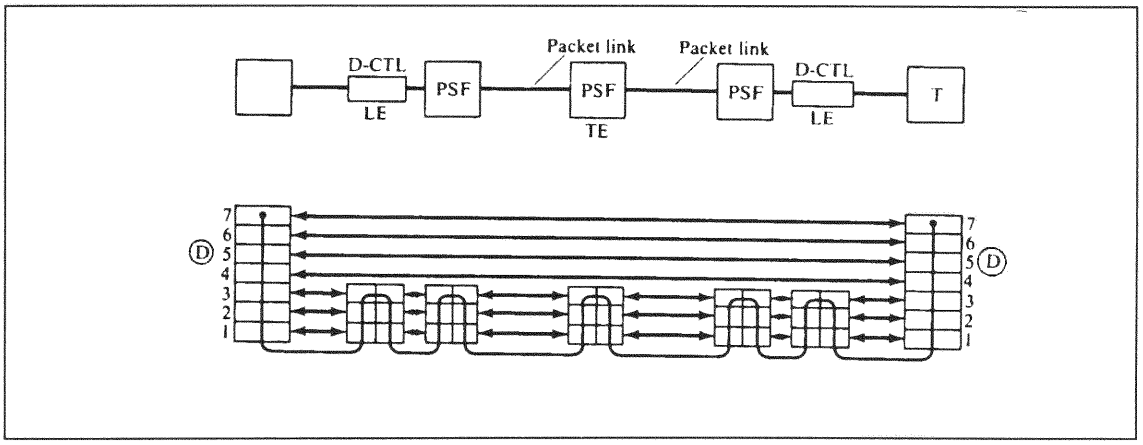


FIGURE 3.10 Network Configuration and Protocols for Packet Switching for D Channel (Courtesy: [Stal92])

3.6 THE ISDN PHYSICAL LAYER

The ISDN BRI S/T reference point specifies digital data transferred in full duplex mode. Two separate physical

circuits are used to achieve full duplex operation, a pair to transmit and a pair to receive. The line coding scheme for this interface is the pseudoternary coding scheme. The pseudoternary coding scheme specifies no signal for a binary one, and alternating positive or negative pulses for binary zero (+/- 750 mv). The pseudoternary coding scheme has the advantage that there is no net dc component, but the disadvantage that a long string of 1's can cause loss of synchronization. FIGURE 3.11-B depicts the transmission structure at the U interface. The transmission structure at the S/T reference point is structured into frames. A frame is depicted in FIGURE 3.11-A and consists of 48 bits at 192 KBPS, so that each frame has a duration of 250 microseconds. Each frame of 48 bits is made up of 16 bits from each B channel, 4 bits from the D channel. The remaining 12 bits are used to synchronize the receiver on the beginning of the frame, dc balancing, and to maintain frame alignment, that is, overcome the limitations of the pseudoternary coding scheme.

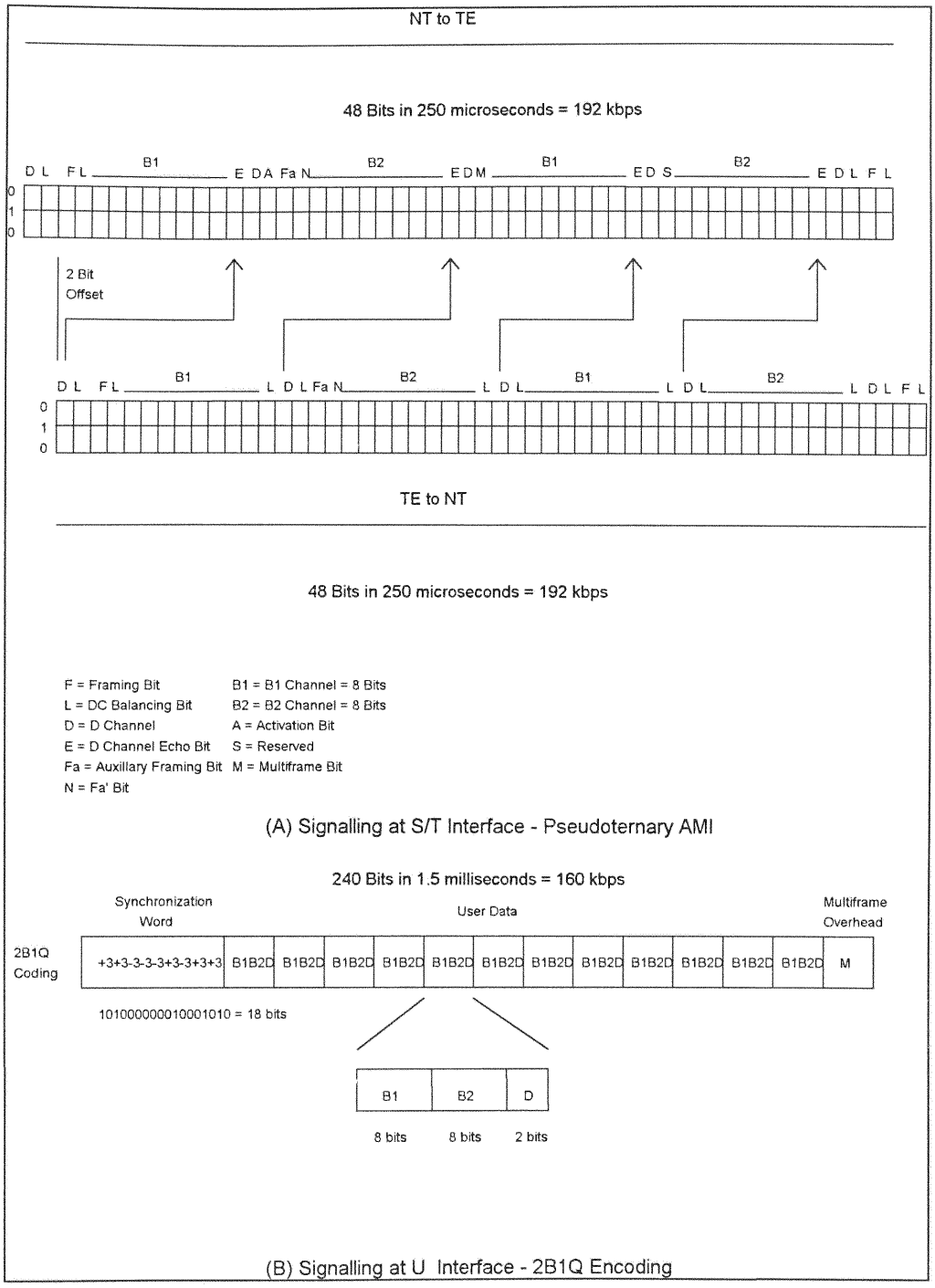


FIGURE 3.11 Transmission Frame at (A)-S/T and (B)-U Interface (Courtesy [Will93])

3.7 THE INTERNET NETWORK

The Internet is a worldwide network system which allows users to communicate with one another. Internetworking is done in the networking layer of the OSI model. The networking layer provides the following:

1. Link between networks.
2. The routing and delivery of data between processes on different networks.
3. An accounting service that keeps track of the use of the various networks and gateways.

There are two kinds of Internetworking approaches: Connection-Mode operation or Connectionless-Mode operation.

Connection-Mode operation assumes that each subnetwork provides a connection-mode form of service, that is, it is possible to establish a logical network connection between two Data Terminal Equipment (DTE) attached to the same subnetwork. Connectionless Mode operation corresponds to the datagram mechanism of a packet switching network.

CHAPTER 4 HARDWARE IMPLEMENTATION OF THE SYSTEM

4.1 MEDNET ARCHITECTURE

MEDNET is a medical networking system that will allow medical personnel to access information about patients regardless of location. The patient or medical personnel can be physically anywhere there is telephone access. In order for MEDNET to be implemented, it is required to have the following:

1. A file server.
2. A client server.
3. A network.

The file server needs to be a multi-user, and a multitasking system. It is essential to be multi-user because it is most likely that several remote and local users would be connected to the system at any time. The file server acts like a kernel. It's program runs in the background mode and it keeps track of incoming users. Once a new user logs in, it requests an attention. The software invokes a new process to respond to the new user's demand. The file server requires a workstation with high

performance, fast throughput and fault tolerance, so that the processing of data will not degrade the performance of the network.

The client server requires a computer with graphics capability such as X-Windows, to interact with an X-Server workstation. The client system must have at least eight megabytes of Dynamic RAM so that it will be able to execute the X-Windows program. The computer needs to have a communication black box such as Northern Telecom (NT) or American Telegram and Telephone (AT&T) ISDN terminal adapter to communicate with the file server.

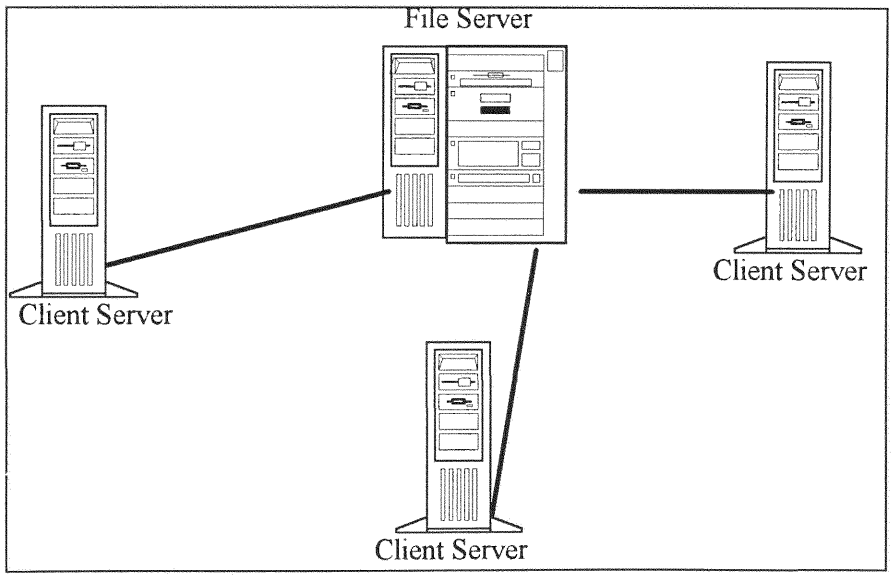


FIGURE 4.1 Multi-user System

An alternative would be a modem, which the client may use to connect to the Internet service provider in order to log into the file server system. FIGURE 4.2 shows a typical client server system.

The network system is used to interconnect the file server and the client server. The connection to the network system accommodated by one of the following:

1. Standard telephone line
2. Internet connection
3. ISDN connection

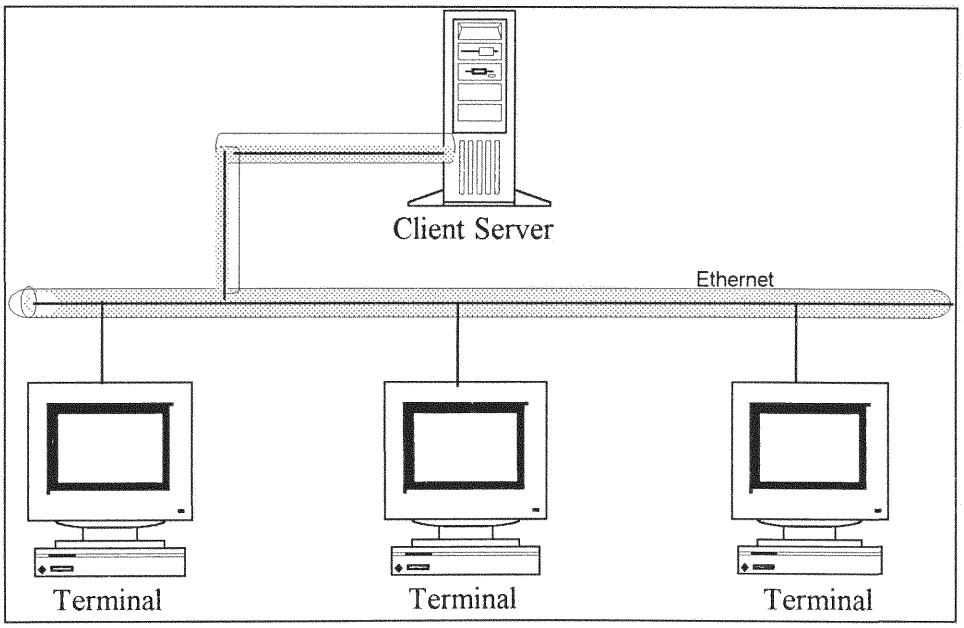


Figure 4.2 Client Server

4.2 THE STANDARD TELEPHONE LINE

Using the standard telephone line requires a high speed modem. Users call directly to the terminal server at the computer center where the file server is located. Once a physical connection is established, the system will direct the line to the file server. The terminal server provides multiple telephone line connections to the main system. FIGURE 4.3 shows such a telephone connected network.

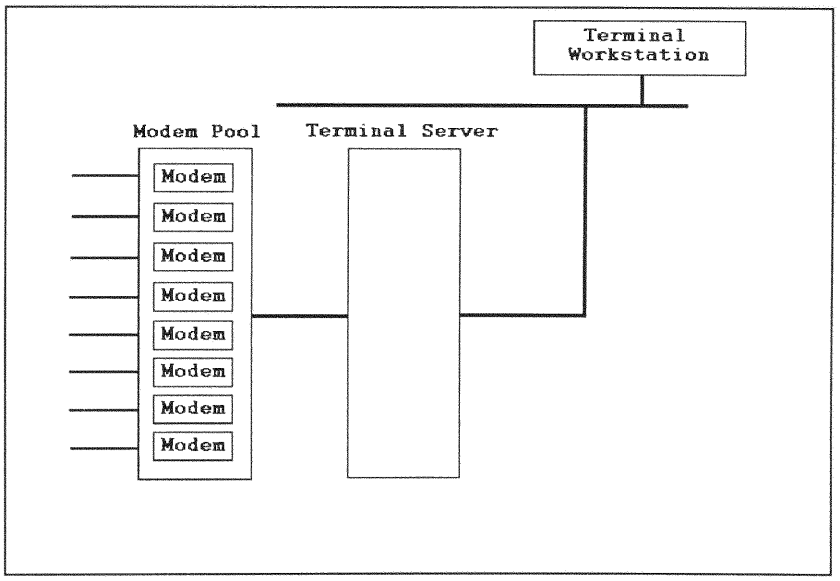


FIGURE 4.3 Standard phone line connection to the network

4.3 THE INTERNET CONNECTION

The file server computer is connected to the Internet network with an RJ45 connector so that the users on the same local network as the server just need to log-in. This user would connect to the server using a graphic program such as X-terminal. Remote users, regardless of location, need to access Telnet to connect to the file server system. Of course, all of these connections are made automatically and are, in most part, transparent to the user. The user connects to the server and logs-in using only a few steps. Using Internet with Xterminal, a user can Telnet to different file servers at same time. This enhances the utilization of the present MEDNET.

When a connection is initiated by a remote site, such as a hospital, the connection is routed through various network nodes until it gets to the destination server. FIGURE 4.4 depicts Internet networking, with a workstation file server and a client Xterminal.

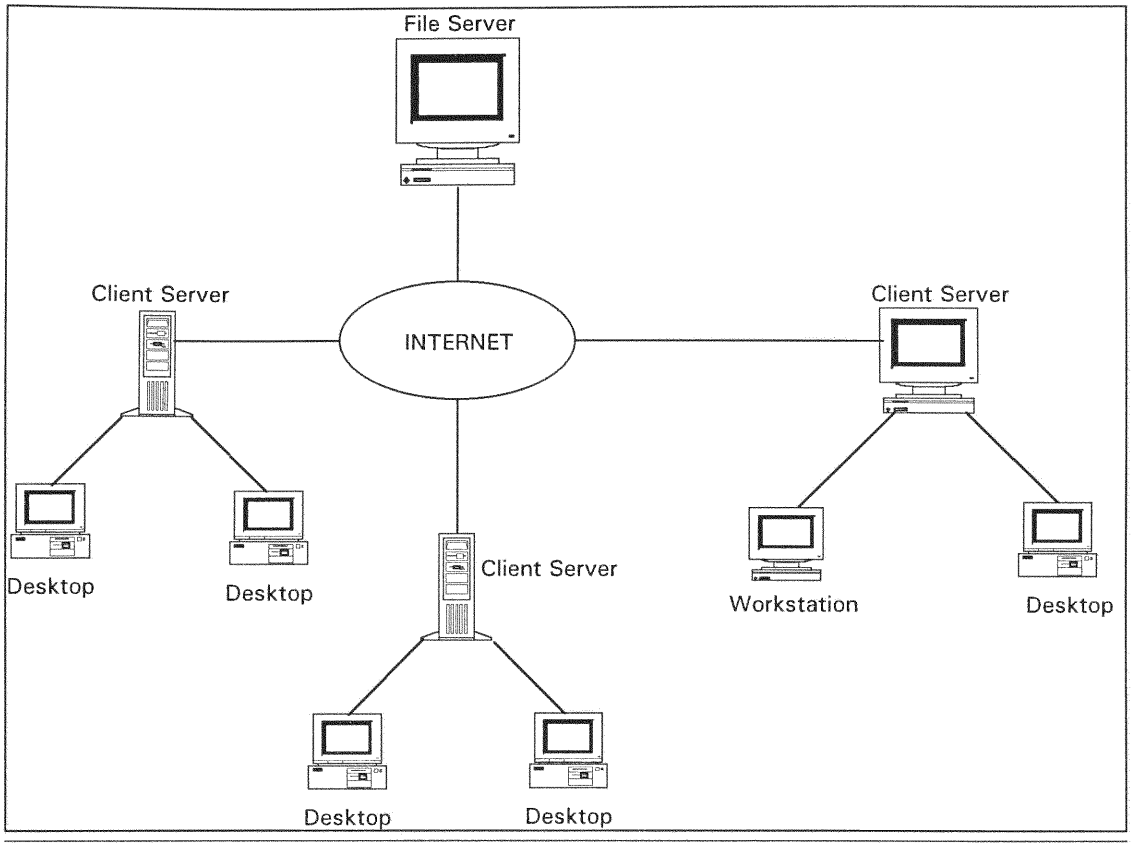


FIGURE 4.4 MEDNET connection

4.4 THE ISDN CONNECTION

The ISDN network is perhaps the most accommodating part of the MEDNET developed at FIU Electrical and Computer Engineering ISDN Lab. The problem with the Internet network is that the user will have to connect to an Internet service provider. Connecting to an Internet service provider limits the user to a maximum transmission rate of 28.8 kbits/sec. Also the Internet Access through a service provider is

expensive. Furthermore, there are federal restrictions on the use of the Internet for commercial purposes [FIPS1026].

4.5 MEDNET IMPLEMENTATION USING ISDN

To design the medical network architecture, ISDN is the most reliable and cost effective communication technology available. In comparison with telephone dial-up and connecting to an Internet service provider, ISDN provides a far more reliable and greater communication bandwith. Today the fastest modem is 28.8 KBPS. ISDN with its end-to-end digital connectivity, can provide a transmission speed of at least 64 KBPS on each B channel. Furthermore, this capacity could be increased by bundling B channels together, that is 2 B channels bundled provide 128 KBPS, 6 B channels bundled provide 384 KBPS, etc.

The ISDN is a perfect fit into the Medical network. The speed and reliability of a of digital line far exceeds the speed of a modem. Medical doctors, for example radiologists, could be on call simultaneously at several hospitals and work more efficiently from one centralized location. High speed ISDN services would be used to transmit this information.

MEDNET uses ISDN in two modes of operation:

1) Circuit Switched Data mode (CSD)
2) Packet Switched Data mode (PSD)

In Circuit Switched Data mode, the system uses the full transmission capability of the B channel of 64 KBPS. If higher bandwidth is required, as in the case of teleconferencing, transferring X-rays or digitized MRI scan pictures, it is easily possible to achieve higher transmission bandwidths through bundling of B channels. FIGURE 4.4 shows how to obtain a transmission bandwidth of 128 KBPS through bundling of two BRI lines [Stal92]. Using the same technique, several B channels can be bundled together to obtain higher transmission bandwidths on demand.

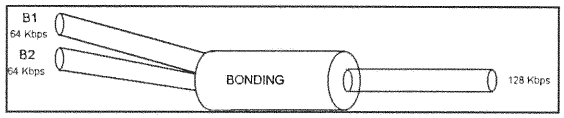


Figure 4.5 Two BRI B-Channels bundled together to provide 128 Kbps

The Circuit Switched Data mode provides a high speed network connection to the file server. Doctors would have to configure the ISDN line for 64 KBPS or 128 KBPS at their site. At the server center, the same configuration has to be made. An interesting fact, is that at the center, the server needs to connect only one ISDN card to provide the same service to more than one ISDN BRI line. FIGURE 4.6 shows the Circuit Switched Data mode connection for the Medical networking configuration.

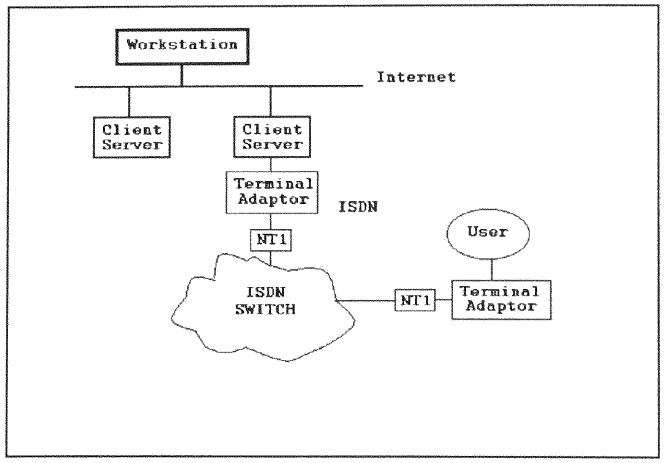


FIGURE 4.6 Circuit Switched Data configuration for the Medical network

The Packet Switched Data mode of ISDN is a very useful mode for networking when cost is the main issue rather than speed of transmission. The ISDN Packet Switch Data mode is very versatile. The transmission rate can be dynamically allocated. When a user subscribes to the ISDN service, the BRI package includes two BINARY (B) channels and a DELTA (D) channel. Each B channel operates at a rate of 64 KBPS. The D channel operates at a rate of 16 KBPS. The D channel carries the signaling information for the associated B channels and allows access to the packet network. This line can be used very efficiently by using it to connect the doctor's workstation to the Medical network in packet mode.

In MEDNET, the file server would have one ISDN card that connects to two BRI lines. Each BRI line is configured as

two packet B and one packet D. The packet B is configured as eight logical channels with an adaptive transmission rate. This gives the file server the capability to serve eight clients at the same time, each at 8 KBPS. If only one user is starting the data transfer using packet B channel, the transmission rate is the maximum allowable bandwidth, that is 64 KBPS. Once more users log-in and start transferring data across the network, the system automatically adjusts the transmission bandwidth by dividing the maximum allowable bandwidth by the number of users. This dynamic bandwidth allocation can continue until the maximum, allowed channels are in use. The transmission rate dynamically readjusts when any MEDNET user is not making any data transfer request. FIGURE 4.7 shows Packet Switched Data configuration.

The usual transmission architecture of the doctor's office is as follows. The doctor's office would subscribe to the ISDN BRI service with two Circuit Switched Voice B, and one Packet Switched Data D. The circuit switched voice is used just like usual voice line. The Packet switched Data channel is used as data line. The doctor's workstation connects to the packet D line at 9.6 KBPS which is very suitable for text transfer. With data compression techniques, data throughput can be much larger that the actual bit rate transfer of the physical medium.

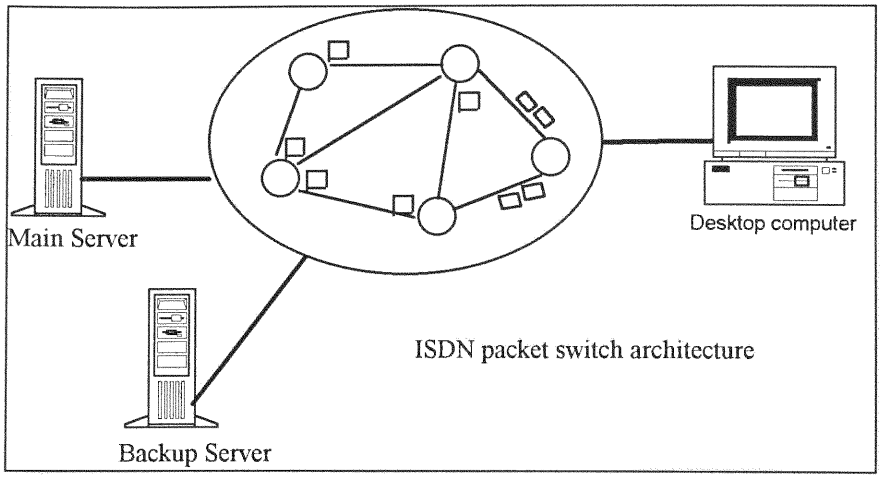


FIGURE 4.7 The file sever connection to the ISDN packet line

At the pharmacy, the file server has one high speed packet B connection, such that, more than one doctor's office can connect and prescribe the medication for their patient. This set up makes it more convenient than the current system for the patient, as was described earlier. It also reduces the possibility of fraudulent alteration and issue of prescriptions.

At home, the radiologist uses the circuit B line to connect to the file servers that are shared by MEDNET users, and also uses the circuit switch data to access the hospitals and radiology centers. This configuration is as designed, because the medical lab and pharmacies do not restrict the access to one user and they use the packet

switch network so that many authorized users could use their service. Since the data transfer is mostly text, the connecting time will not be long and transfer speed provided is sufficient. A Circuit Switched connection must be used for direct connection to the hospital and radiology centers for receiving X-rays and other medical imaging files. These imaging files are generally large files, and thus require the full bandwidth of the B channel.

Doctors could use the Circuit Switch Data service to connect to other doctors for consulting as a second opinion. Other specialists that do not require X-ray or medical imaging transferring facilities, may still have a need to connect to the file server at the clinic to download patient information. This configuration would require the use of a packet B channel to connect and retrieve the patient file from the server system, which could be done simultaneously while conversing to the patient over other B channel.

CHAPTER 5 SOFTWARE DEVELOPMENT

MEDNET software is written in the C++ language and is compiled with the GNU g++ compiler. The program consists of 12 modules listed below:

1) *dstart.C*
2) *pstart.C*
3) *lstart.C*

4) *pstartb.C*
5) *dstartb.C*
6) *lstartb.C*

7) *pharmacy.C*
8) *doctor.C*
9) *lab.C*

10) *dserver.C*
11) *lserver.C*
12) *pserver.C*

5.1) DSTART PROGRAM

When the doctor logs in to the client server, the program *dstart* is invoked by the .loginfile, and spawns a child

process to initiate the remote connection to the main file server by executing the *rsh* command. The parent process of the *dstart* program then monitors the validity of the connection by *pinging* the main file server at the remote site. If a "no response" message is received, the parent process terminates the child process, invokes the program

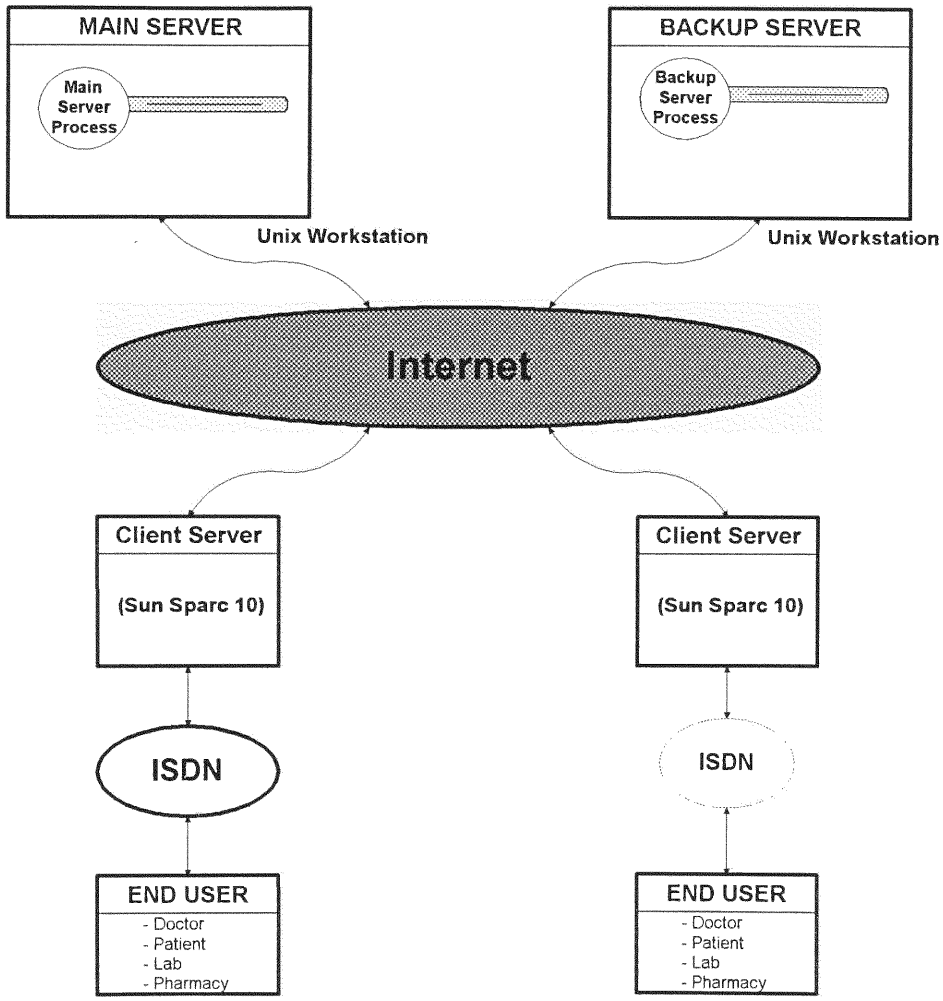


FIGURE 5.1: Shows the user connection to file server through client server.

called *dstartb*, and terminates itself. *Dstartb* spawns a child process to reinitiate the remote connection, but this time to the backup file server by executing another *rsh* command. The parent process *dstartb* monitors the validity of the connection by pinging the backup file server at the remote. The source listing of the module is included in Appendix A.

Figure 5.1 shows the user end, main file server, and backup server connection to Internet. The diagram illustrates how the user, in this case a doctor or pharmacist, needs to connect to the client server in order to gain access to the main file server.

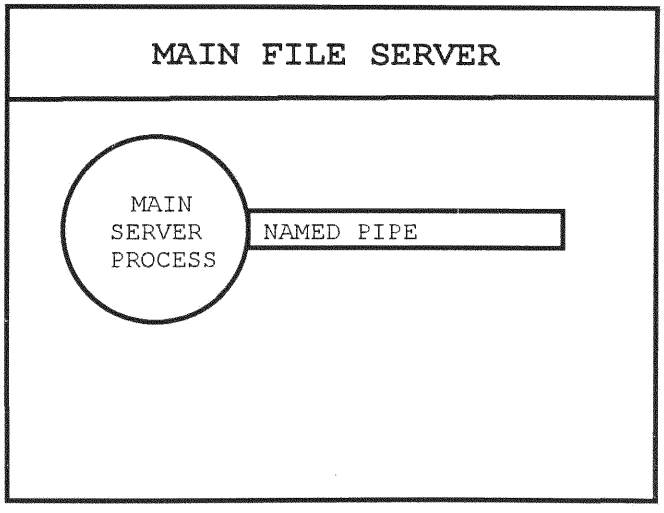


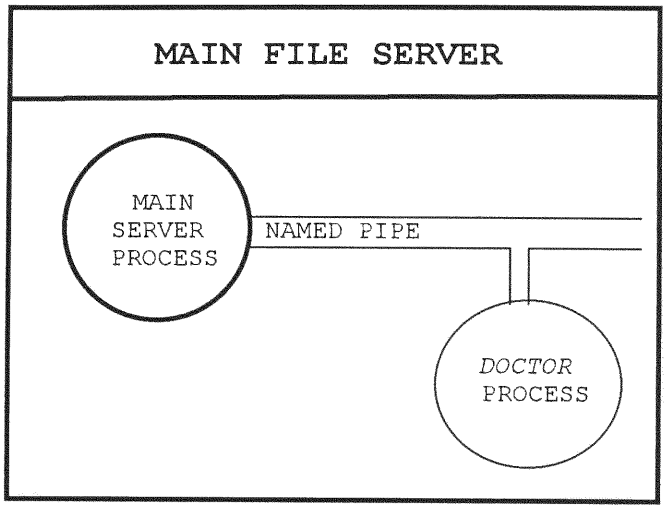
FIGURE 5.2: File server running main server program with named pipe open.

5.2 DOCTOR PROGRAM

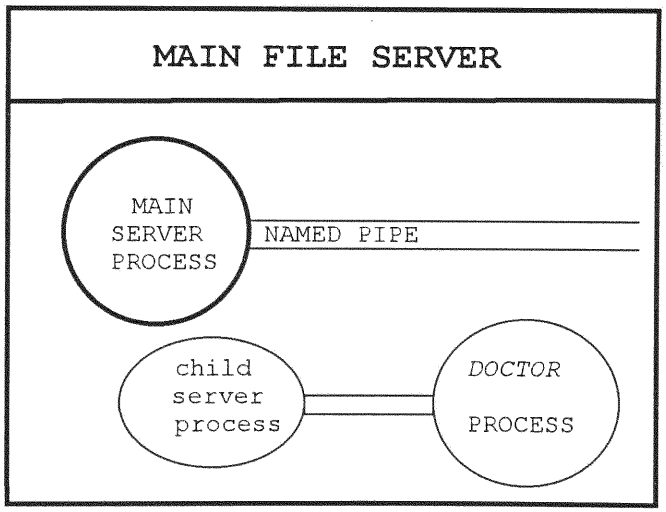
The *doctor* program is invoked by the *dstart* program from the remote client server. This module is responsible for displaying the option menu and providing the user interface to the file server. Once the *doctor* program is invoked, it will start the communication with the *dserver* module already running at the file server as a background process. FIGURE 5.2 shows the main file server running the *dserver* with the named pipe opened.

Dserver has two named pipes which are opened for reading, writing, and monitoring the pipe for any newly invoked clients. The first step for the doctor module to communicate with the file server is to use it's own process id to create and open two named pipes by concatenating letter `a' and `b' to the process id. For example, if the pid of the doctor process is 1234, the letters `a' and `b' are added to create the two named pipes for reading and writing, and the named pipes become 1234a and 1234b respectively. The next step for the *doctor* module is to send it's process id to *dserver* module. Once the *dserver* module receives the doctor's process id, it spawns a child process to handle the *doctor* modules request. The *dserver* child process uses the *doctor* process id to create the name of the

two named pipes that are already made by the *doctor* process by concatenating the letters `a' and `b' to *doctor* process



(a) shows *doctor* communicating with main server.



(b) *doctor* communicating with child process.

Figure 5.3: *doctor* communication with server.

id. The next step for the child process is to open the two named pipes for starting the communication process, Figure 5.3 shows *doctor* communicating with main server, and *doctor* communicating with child process.

Once the child opens the two named pipes, *doctor* starts communication.

The *doctor* module provides the following options to user:

-add a record

-find a record

-print a record

-delete a record

5.2.1 ADD A RECORD

To add a record, the program opens a file at the doctors display, so the doctor can fill in the proper information such as patient's personal information and patient's health related information. When the patient's record properly is written, the program saves the record as a file with the social security as the file name. The program also adds the patient's name with the patient's social security number to a file called ptrfile. *Ptrfile* is a file that contains the list of all patients.

5.2.2 FIND A RECORD

To find a record, the *doctor* program prompts for the name of the patient from the user and when entered, sends it to *dserver* module through the pipe. The d*server* module makes a binary search of the patient list and returns the name of the file that contains the patient's record. The doctor module opens the file and displays the patient's record at the doctors display with the patient picture. At this point, the doctor will be asked if he/she wishes to open a prescription file or to close the file and return to the main menu. If the doctor prescribes a medicine, after closing the file, the prescription record is appended to the patient's record.

5.3 DSERVER PROGRAM

Dserver module is responsible for maintaining the patient list in a balanced binary tree data structure for fast data retrieval. Once the dserver program is invoked, it loads the patient list into the binary tree and maintains the balance of the tree. The dserver then creates and opens the two named pipes for communication with the incoming clients, and monitors the pipe. When a new client process is invoked, it will first write it's process id into the named pipe that has been created by the dserver and then it makes two named

pipes using it's process id with letters `a' and `b' concatenated to it.

As soon as the dserver at the other end reads the named pipe, the dserver spawns a new child server process. The newly born child process then creates the name of the pipes that are already created by the client process, and opens the pipes for inter-process communication. After the connection is established, the next step is for the child process to look at the pipe and wait for a request to be made by the client process. If the client makes a request, the server process will do a binary search of the patient list and return to the client the file name that contains the patient record. The child server process stays alive and active as long as the client process is alive. Once the client process is terminated, the child server process also terminates automatically. Therefore, no zombies will remain at the file server.

CHAPTER 6 SYSTEM INTEGRATION

6.1 HARDWARE AND SOFTWARE INTERFACES

The hardware and software interfaces of the MEDNET system provide for a secure and user friendly access to patient data on the Medical Network.

In the FIU Lab, MEDNET is implemented using ISDN and Internet communication technologies. The concept of the MEDNET system is portable to different network platforms. The hardware model consists of a SUN Sparc 10 workstation, a UNIX base computer with 64 Mg bytes RAM, a one Gigabyte hard drive, an NCD X-terminal, and PC-compatible computers with X-terminal software running under the Windows operating system. The SUN Sparc comes with an Internet interface card with an RJ45 connector. It also comes with built-in AT&T ISDN Terminal Adapter (TA). The Terminal Adapter of the file server workstation connects to a Network Terminator (NT), which in turn is connected to the ISDN switch at the Central Office (CO).

On the other side of the ISDN line the user's system is connected to an ISDN switch in a similar fashion. Users of the MEDNET could be served by different Central Office switches. For example the file server could be connected to

a Northern Telecom DMS-100 switch while one user could be connected to an AT&T 5ESS switch and another user connected to Siemens EWSD switch. FIGURE 6.2 shows the complete connection of two users to the server through ISDN.

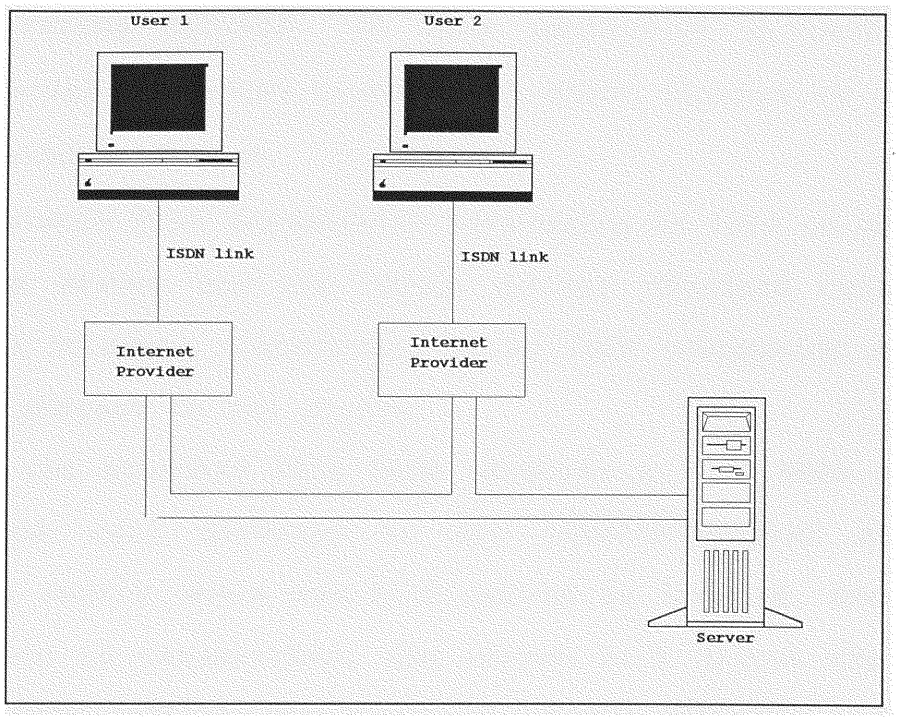


Figure 6.2 Complete connection of 2 users to the MEDNET server through ISDN

The operating software was designed for dynamic data base implementation, and implemented using GCC, the C++ compiler. For fast data access, the Binary-tree structure with automatic B-tree balancing algorithm is used. On the remote

site the PC computer runs with X-terminal software under Windows, NCD X-terminal and Sun3/80.

The development guide graphical user interface is used for the development of the graphical interface of the MEDNET system. It supports the development of portable MEDNET system applications that can run on a variety of hardware and operating systems. In this particular project, the system model is tested using a Sun Sparc Station 10 on top of a X-windows system.

The Network File Server(NFS) networking is used to connect the file server and the users. Either Standard telephone line, an Internet connection, or an ISDN connection can be used to connect to the system.

Using the standard POTS line requires a 14.4 KBPS modem. Users can call directly to the terminal-server at the computer center where the file server is located. Once the connection is established, the system will direct the call to the file server. The terminal server provides multiple telephone access to the main system.

The workstation is connected to the Internet network. Users that desire to log-in need to connect to an Internet services provider, and Telnet to the system with an X-terminal graphic program.

When a call is initiated by a remote user from a place like a hospital, the call is routed through various network

nodes until it gets to the destination MEDNET server. FIGURE 6.3 shows the use of Internet to access the file server from client X-terminal.

ISDN is the best communication technology available to implement the MEDNET medical network. In comparison with standard POTS dial-up at the maximum rate of 28.8 KBPS to interconnect an Internet service provider, ISDN is much

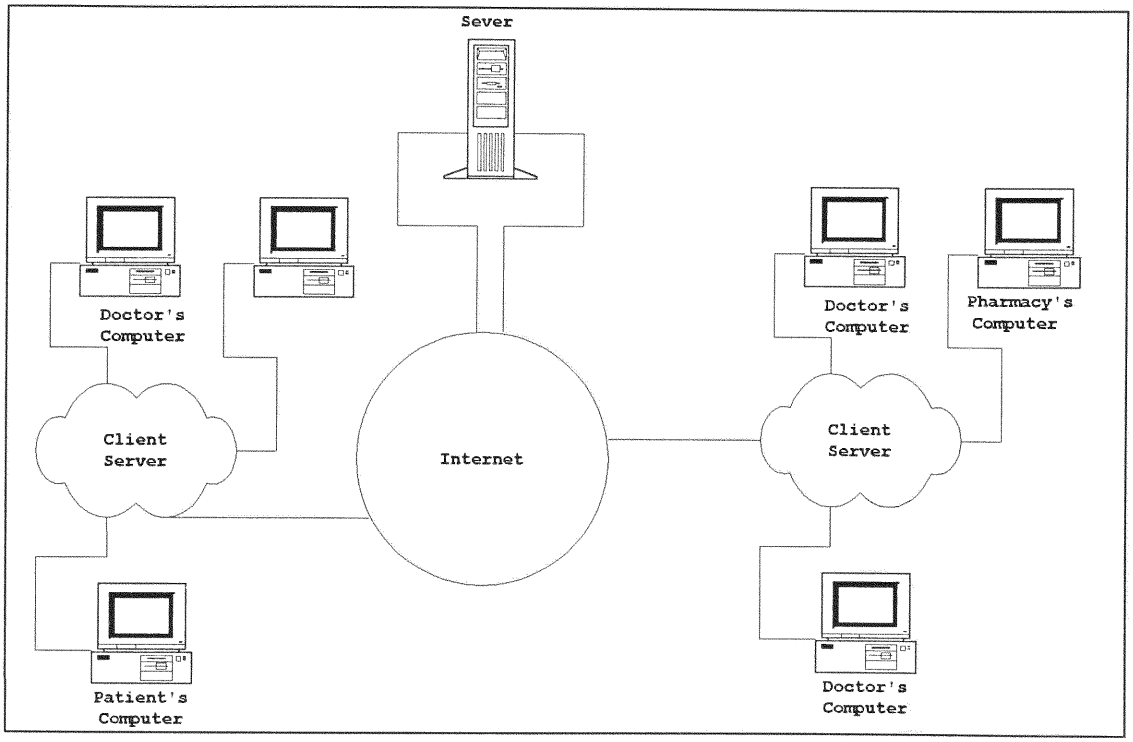


FIGURE 6.3 Internet to access a file server

faster and more reliable. ISDN with it's end to end digital from the client X-terminal connectivity offers higher reliability and a speed of transmission of 64 KBPS per B channel. Medical doctors, like radiologists, could be on call at than more one hospital, and work more efficiently from one centralized location using high speed ISDN services for image transfer. MEDNET uses ISDN in both operations modes: CSD and PSD. In either mode, the system uses the full speed of 64 KBPS per channel. Furthermore, B channels could be bundled together as shown in FIGURE 6.5, to provide higher transmission bandwidth for medical imaging.

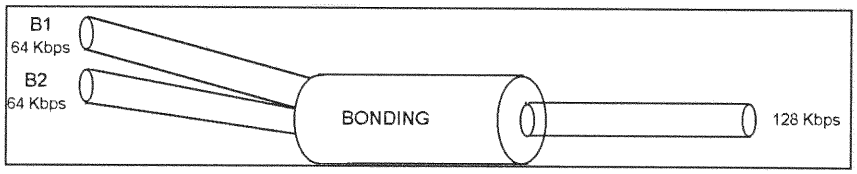


FIGURE 6.4 Two BRI channels bundled to provide 128 KBPS

6.2 THE MEDNET FILE SERVER

The MEDNET file server has one ISDN external terminal adapter that interfaces to one B channel. FIGURE 6.6 shows the file server connection to ISDN.

The communication architecture of the doctor's office is to subscribe to the ISDN BRI service with two CSV B channels, and one PSD D channel. The CSV channels provide for voice access, and the data channel provide access to MEDNET. FIGURE 6.5 shows the communication architecture of the doctor's office.

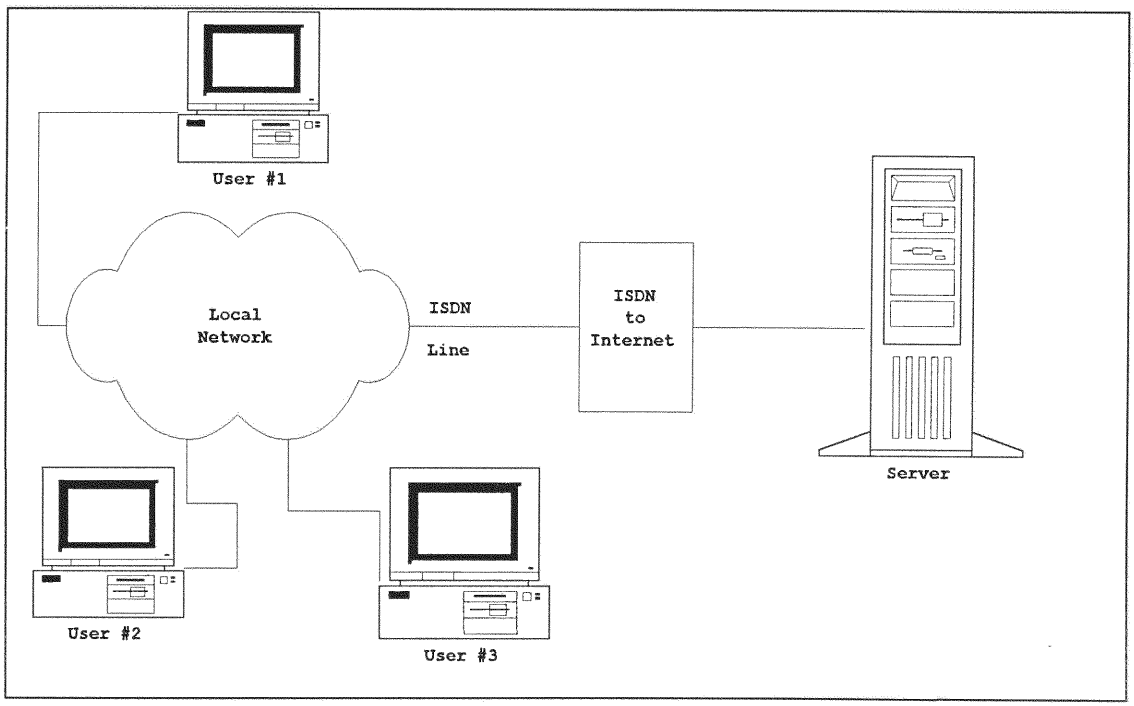


FIGURE 6.5 File server connection to ISDN

6.3 MEDNET INTEGRATION

The integration of the MEDNET system allows medical personnel to communicate without the need for voice communication. In another words, once the user, such as the doctor, is in the MEDNET system, he/she will be able to call the pharmacist through the MEDNET system so that the two of them can converse through the network. Meanwhile, communicating parties' monitors display a picture of each another for validation purposes. Although one might presume this process would be time consuming, with ISDN in place, the communication delay is negligible to the user.

6.4 MEDNET CASE STUDY

The MEDNET system can be initiated by the patient's call to the doctor. The doctor enters the name of the calling patient on the system and instantly the patient's picture appears on the doctor's monitor. Also, the doctor's picture appears on the patient's monitor. If for any reason the doctor needs to call the medical laboratory to obtain a lab result for the patient, then the doctor's and the patient's picture display on the laboratory's computer. Also, the laboratory personnel's picture is displayed on the doctor's

monitor. Once the doctor's request has been processed by the lab personnel, the doctor would disconnect from the medical lab. The lab personnel's picture is erased from the doctor's monitor. Once the doctor has received the patient's lab results and has formulated a diagnosis, then the doctor calls the pharmacist to provide all the necessary medication information. While conversing with the pharmacist, the doctor's and the patient's pictures are displayed at the pharmacist's terminal. At the same time the doctor has a picture of the pharmacist on his/her screen. The idea is that all the medical personnel are able to access each other's image and medical information (both text and image) while conversing through the MEDNET system.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

MEDNET is a comprehensive medical networking system which can interface to standard POTS analog modems, Internet, and primarily the Integrated Services Digital Network (ISDN). MEDNET is a network that can interconnect healthcare providers, such as clinics, hospitals, private and state medical offices, pharmacies, and insurance companies into a single system which allows all qualified personnel to access a patient's data file in near real time. FIGURE 7.1 is a system diagram of MEDNET.

The MEDNET system is implemented using a UNIX based system, Sun Spark 10, as a file server. A user can access the file server through either analog modem,

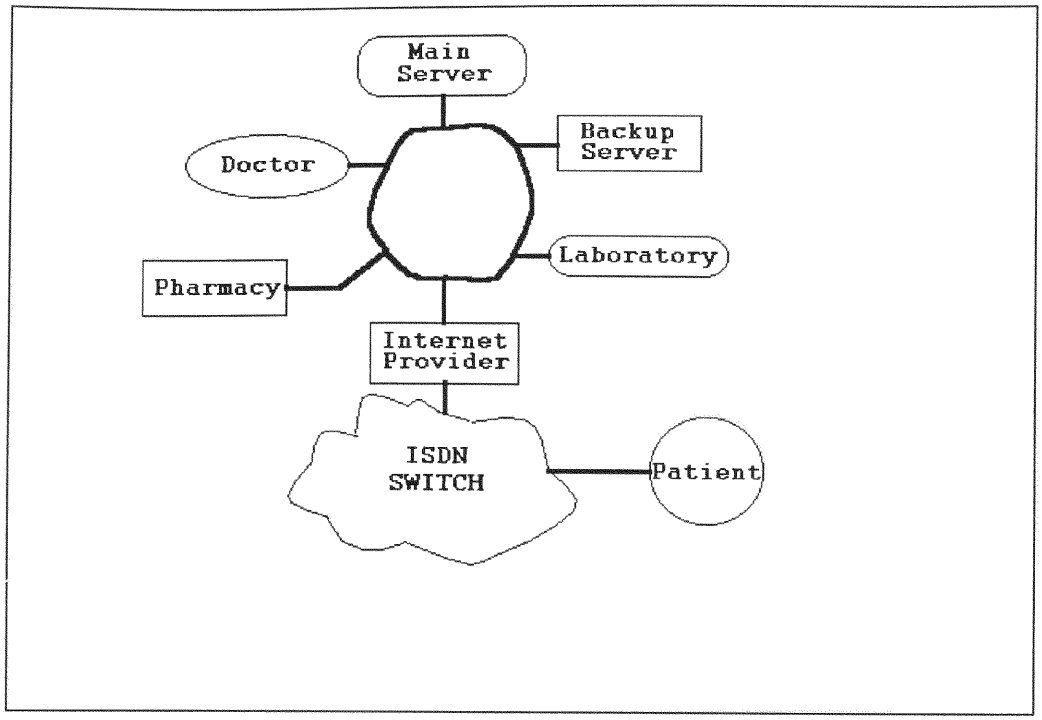


FIGURE 7.1 MEDNET System Diagram

Internet or ISDN. In the case of modem access, the user may obtain access to the file server by dialing a telephone number to be connected to the terminal server. As for Internet access, the user can Telnet to the file server from any machine that has Internet connection capability. Finally, the user may use an ISDN line to interconnect to the file server.

Currently, the maximum baud rate of a modem is 28.8 KBPS, or sometimes in ideal circumstances it can not exceed 32 KBPS. As for Internet, the connection requires an expensive communications line (T1) and equipment. Furthermore, if a

user doesn't want to invest in a T1 line, he/she would be restricted to use a modem to access an Internet service provider. Therefore, the baud rate to communicate between an individual and the file server would be the same as a that of a POTS standard modem. In view of the current technologies, ISDN seems to be the best solutions to intercommunicate in the MEDNET system.

At the present time, ISDN is readily available to the general public in most areas. Basic Rate Interface(BRI) provides two B channel and one D channels. Each individual B channel has a data rate of 64 KBPS. On the other hand, the D channel has a data rate of 16 KBPS. To obtain a higher bandwidth for data transfer, the two B channels can be combined or bundled through the Terminal Adapter.

ISDN is considered to be more effective as compared to using a POTS analog modem or Internet because it allows healthcare providers to access MEDNET in near real time in a cost effective and reliable manner.

7.2 FUTURE WORK

The ISDN packet network can be utilized for further enhancements of MEDNET system. The enhancements can include the following ideas:

1) Data encryption/decryption.
2) Patient insurance card verification.
3) Patient data file card (Chip Card).
4) Statistical analysis software with multicasting.
5) Fault tolerance.
6) Data Compression.

First, data encryption/decryption can be used in MEDNET to protect from unauthorized users. There are several ways of implementing a cryptography system. Currently, the two most popular algorithms used for data encipherment are the Data Encryption Standard (DES) secret key algorithm, and the Rivest-Shamir-Adleman (RSA) and Diffie-Hellman public key algorithms. The DES algorithm encrypts 64 bits of plain text using a 64 bit key. The 8th bit of each byte of the key is used for parity, which reduces it to a 56-bit key. Then the bits in the plain text are scrambled through a series of substitutions and permutations. The DES algorithm has resisted 15 years of cryptographic attack. RSA is the best known public key cryptosystem. It requires each user to have a pair of keys: a private key and a public key. The

theory behind the RSA algorithm is that the sender will encrypt a message using the receiver's public key. Upon receipt of the encrypted message, the receiver will decrypt the message using his/her own private key. This algorithm works due to the mathematical nature of the public and private keys, which are large, at least 508 bits, and must be able to be factored into prime numbers. Therefore, MEDNET system can choose either DES, RSA or a Hybrid algorithm [Fern94] to secure the system. For example, the patient files could be encrypted so that the staff personnel would not be able to access patient's medical records.

The second proposed future enhancement of MEDNET is the use of a patient insurance card which includes insurance verification information. This information is important to the MEDNET system for the purpose of authentication. Basically, the insurance verification information could be a stored in the card's magnetic stripe. The insurance information can be stored in three tracks. The most important data includes the group number, the patient's social security number, and an identification number so that all medical offices and pharmacies can verify the patient insurance policy through the ISDN network. FIGURE 7.2 depicts such Insurance Verification Card.

The third proposed future enhancement is the Patient Data File card (Chip Card). The Chip Card can be used in MEDNET to simplify the entire system. The patients can have their past medical records stored in a chip card to save time and convenience for the medical staff. There are many kinds of chip cards. The two major types are memory cards and smart cards. Memory cards can store a limited amount of

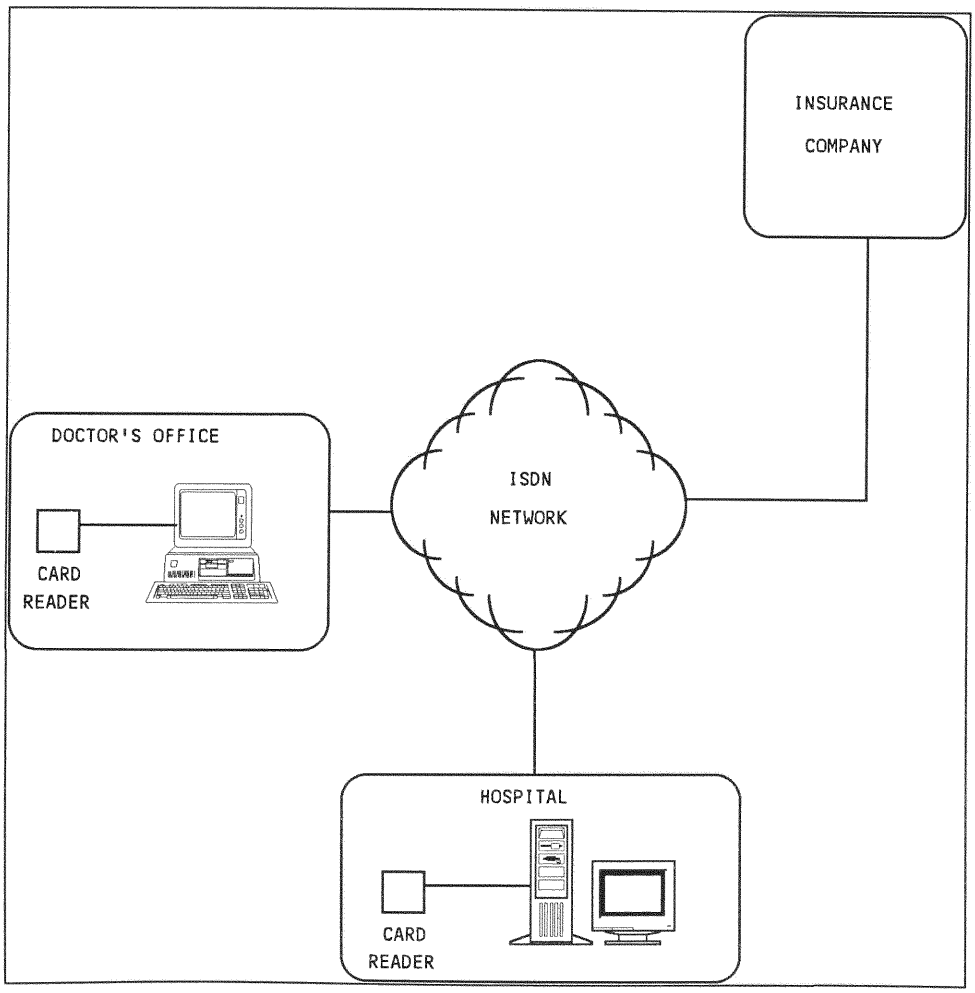


Figure 7.2 Insurance Verification Card

information and it has little or no security features. Basically, if an unauthorized user gets hold of a patient's card, it would be easy for the hacker to obtain all the stored medical records. On the other hand, a smart card is expensive but it has more memory and higher security. Most smart cards in today's technology have data encryption/decryption built in. To obtain access to a certain part of the memory, a patient needs to provide the key to encrypt or to decrypt the data and obtain access. For example, SGS Thomas manufactures smart cards that need a key or keys to obtain different levels of access. FIGURE 7.3 shows the use of a Chip Card reader in MEDNET.

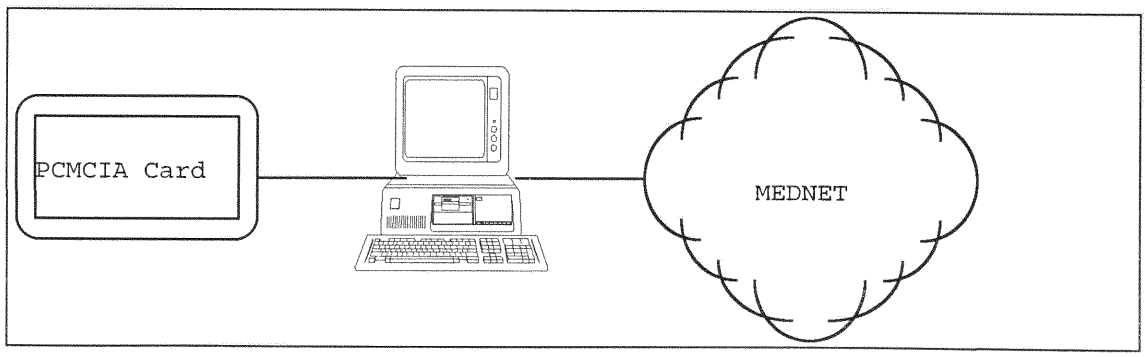


FIGURE 7.3 Chip Card reader with MEDNET

The fourth proposed future enhancement is statistical analysis software with multicasting which is beneficial to the MEDNET for research purposes. Since the patient's medical history and current records are in

the file server, it would be possible for some medical researchers to gain access to only certain fields of the records so that the privacy of the patient is preserved and at the same time the medical records are available for statistical analysis on certain diseases such as alcoholism, liver cancer, lung cancer, drug addiction, etc.

The fifth proposed future enhancement is Fault Tolerance techniques which is an important enhancement to Medical Networking. One technique is by having multiple copies of the file in a single or multiple file servers in order to protect the patient data from being damaged or destroyed through man/machine errors.

The sixth proposed future enhancement is data compression implemented in the medical network to enhance the system in speed and storage utilization. Since the processor's speed is much greater than the transfer rate of the ISDN line, the communication system can become a bottle neck. In other words, by compressing the data before transmission not only the transmission gains speed but reduces the file storage requirements.

LIST OF REFERENCES

[AbrAL93] M. Abrams, D. Gambel, S. Jajodia, H. Podell, and R. Sandu, Course notes "Recent Developments In Information Security, Information Security Institute, George Mason University, October, 1993.

[AgnAL90] G. Agnew, R. Mullin, S. Vanstone, "Common Application Protocols for the CA34C168 and their Security Characteristics", Newbridge Microsystems CMOS Products Databook, 1990.

[AgnAL88] G. Agnew, R. Mullin, S. Vanstone, "A secure public key protocol based on discrete exponentiation", Proceedings of Eurocrypt 88, Springer Verlag.

[BetAL91] Th. Beth, M. Frisch, and G.J. Simmons, "Public-Key Cryptography: State of the Art and Future Directions", EISS Workshop, July, 1991.

[BihSha90] E. Biham and A. Shamir, "Differential Cryptoanalysis of DES-like Cryptosystems", the Weizmann Institute of Science, Technical Report CS90-16, July 1990.

[Burr91] W. Burr, "Security in ISDN" - NIST publication, March 91.

[ComSec] "Computer Security Products Buyers Guide 1993", Computer Security Institute, 1993.

[ClaKuh88] G. J. Claassen and G.J.Kuhn, "Secure Communication Procedure for ISDN", COMSIG 88 Pretoria, pp.165-170.

[DagAL94] N. Dagdeviren, J. Newell, L. Spindel, and M. Stefanick, "Global Networking with ISDN", IEEE Communications, vol.32, no.6, pp.26-32, June, 1994.

[Davi81] D. Davies, "The Security of Data in Networks", IEEE Computer Society Press, 1981.

[Data93] Data I/O Corporation, 10525 Willows Road N.E., P.O.Box 97046, Redmond, WA 98073-9746.

[DavPri80] D. Davies and W. Price, "The Application of Digital Signatures Based on Public Key Cryptosystems", Proceedings of the Fifth International Computer communications conference, pp.525-530, October, 1980.

[Denn82] D. Denning, "Cryptography and Data Security", Addison-Wesley, 1982.

[Denn93] D.Denning, "The Clipper Chip: A Technical Summary", April 19, 1993.

[DifAL89] W. Diffie, B. O'Higgins, and S.Schnider, "Secure Communications", Telesis, pp. 42-50, 1989.

[DifHel76a] W. Diffie and M. Hellman, "New Directions in cryptography", IEEE Trans. Inform. Theory vol. 22 , no. 6, pp. 644-654, 1976.

[DifHel76b] W. Diffie and M. Hellman, "A critique of the proposed data encryption standard", CACM, pp.164-165, March 1976.

[DifHel77] W. Diffie and M. Hellman, "Exhaustive Cryptoanalysis of the NBS Data Encryption Standard", Computer, vol.10, no. 6, pp.74-84, June 1977.

[DifHel79] W. Diffie and M. Hellman, "Privacy and Authentication: An Introduction to Cryptography", Proceedings of the IEEE, vol. 67, no. 3, pp.397-427, March, 1979.

[DinKuh93] C. Dinkel and D. Kuhn, "Telecommunications Security Guideline", NIST Technical Report, April, 1993.

[Durr94] M. Durr, "ISDN Reemerges", LAN Magazine, pp.103-112, January, 1994.

[Dwor91] F. Dworak, "Approaches to detecting and resolving feature interactions", Proceedings of the IEEE, IEEE Globecom, 1991.

[EhrAL78] W. Ehrsam, S. Matyas, C. Meyer and W. Tuchman, "A cryptographic key management scheme for implementing the Data Encryption Standard", IBM Systems Journal, vol.17, no. 2, pp.106-125, 1978.

[ElGa85a] T. El Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, vol.IT-31, no.4, pp.469-472, 1985.

[ElGa85b] T. El Gamal, "A Subexponential-Time Algorithm for Computing Discrete Logarithms over GF(p)", IEEE Transactions on Information Theory, vol.IT-31, no.4, pp.469-472, 1985.

[Elme94] P. Elmer-Dewitt, "Who should keep the keys", Time Magazine, pp. 90-91, March, 1994.

[FeiAL75] H. Feistel, W. Notz, and J. Smith, "Some Cryptographic Techniques for Machine-To-Machine Data Communications", Proceedings of the IEEE, vol. 63, no. 11, pp. 1545-1554, November 1975.

[FIPSESS] "Public record concerning the proposed Federal Information Processing Standard for an Escrowed Encryption Standard (ESS)", National Institute of Standards and Technology (NIST), Department of Commerce, Docket No.930659-3159, RIN 0693-AB19.

[FIPSP46] "Federal Information Processing Standards Publication 46 - The Data Encryption Standard", National Bureau of Standards, Government Printing Office, January, 1977.

[FIPS1026] Federal Information Processing Standards Publication 1026 - Telecommunications: Interoperability and security requirements for use of the Data Encryption Standard in data communication systems", National Bureau of Standards, Government Printing Office, Government Printing Office, 1980.

[Folt83] H. Folts, "OSI Workbook", VA:OMNICON, 1983.

[Ford92] W. Ford, "Security Techniques for Network Management", NOMS 92, pp.22.2.1-22.2.17.

[Good92] M. Goodwin, "Guide to Serial Communications", 1992.

[Grif90] J. Griffiths, "ISDN Explained Worldwide Network and Application Technology", John Wiley & Sons, 1990.

[Haye90] Hayes ISDN PC Adapter , ISDN Adapter Card for the IBM PC, XT, At, PS/2 Model 30 or 100% Compatible, Technical Reference, no. 98-00827 AA K10, 1990.

[Hell80] M. Hellman, "The Mathematics of Public-Key Cryptography", Scientific American, vol. 214, no. 2, pp. 146-157, August, 1979.

[IEP1157] P1157 Standard for Healthcare Data Interchange-Overview and Framework, IEEE Draft 3/30/93.

[IIW91] "ISDN Security Architecture" Draft, IIW/S/91-A001, Sept. 5, 1991.

[Impr84] S. Improta, "Privacy and Authentication in ISDN: the Key Distribution Problem", ISS '84 Florence, pp.7-11, May 1984.

[ISO7498] ISO 7498: "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture", February 1989.

[ISO8372] ISO 8372 "Information Processing-Modes of operation for a 64-bit block cipher algorithm", 1987.

[ISO9798] ISO 9798-2 "Entity authentication using symmetric techniques", July 6, 1990.

[Jack90] K. Jackson, "Secure Information Transfer-PC Encryption: A Practical Guide", CRC Press, 1990.

[JanMol91] P. Janson, and R. Molva, "Security in open networks and distributed systems", Computer Networks and ISDN Systems, vol 22, pp.323-346, 1991.

[John92] P.Johnson, "Security and Security Management - Overview of Concepts, Standards Status and Some Current Issues", NOMS, pp.22.1.1-22.1.10, 1992.

[Katz77] H. Katzan, "The Standard Data Encryption Algorithm", Petrocelli Books, New York 1977.

[Kimm92] J. Kimmins, "Network Security Management and Administration: Concepts and Issues", NOMS 92, pp.22.3.1-22.3.12.

[Klue92] H. Kluepfel, "A systems engineering approach to security baselines for SS7", Bellcore Technical Report, TM-STS-020882, 1992.

[Leva94] C. Levay, "Computer Simulation Study on the Impact of Express Lanes on the Current Toll Plaza System", M.S. Thesis, Florida International University, April, 1994.

[Levy94] S. Levy, "The Cypherpunks vs. Uncle Sam", The New York Times Magazine, pp.44-60, June,1994.

[Merk78a] R. Merkle, "Secure Communications Over Insecure Channels", Communications of the ACM, vol. 21, no. 4, pp.294-299, April, 1978.

[Merk78b] R. Merkle and M. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks", IEEE Trans. on Info. Theory, vol. IT-24, no. 5, pp. 525-530, 1978.

[Merk79] R. Merkle, "Secrecy, Authentication, and Public Key Systems", Ph.D. Thesis, Stanford University, 1979.

[Miku93] D. Mikula, "Secure Group III Facsimile Transmission through ISDN" NIUF ISDN Application Analysis 050016.0, 1992.

[Monk92] T. Monk, "Window's Programmer's Guide to Serial Communications", SAMS Publishing, Indiana, 1992.

[NeeAL78] R. Needham and M. Schroeder "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, vol.21, no. 12, pp.993-999, December, 1978.

[Newb92] Newbridge Microsystems, CMOS Products databook, Calmos product line, Issue 3, Chapter 4, 1991.

[NIUF94] North American ISDN Users Forum (NIUF), "A catalog of National ISDN Solutions for Selected NIUF Applications, Second Edition, Section 3, Feb. 94.

[NRC89] National Research Council, "Growing Vulnerability of the Public Switched Networks", National Academy Press, 1989.

[NSTAC90] National Security Telecommunications Advisory Council, "Report of the Network Security Task Force", National Security Telecommunications Advisory Council, 1990.

[OHiAL87] B. O'Higgins, W. Diffie, L. Strawczynski, and R. de Hoog, "Encryption and ISDN - A Natural Fit", IEEE International Switching Symposium 1987, March 15-20, 1987, Phoenix, Arizona, pp.A11.4.1-7.

[OyaAL92] I. Oyaizu, K.Tanaka, and K. Miyabo, "Multipoint Audiographics Teleconferencing System with Privacy Feature", Proceedings of the SPIE Visual Communications and Image Processing 92, Vol. 1818, pp.1489-1501.

[Pere92] P.Perez, "A Simulation of Vehicle Energy Consumption in a Metropolitan Rail Transit System", M.S. Thesis, Florida International University, November,1992.

[Prit80] J. Pritchard, "Data Encryption", NCC Publications, Manchester 1980.

[RivAL78] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, vol 21, no 2, pp.120-126, February 1978.

[RSAL92] RSA Laboratories "RSAREF: A Cryptographic Tool kit for Privacy-Enhanced Mail, Library Reference Manual", March 1, 1992.

[ShaAL78] A. Shamir, R. Rivest, and L. Adleman, "Mental Poker", MIT Laboratory for Computer Science, Report TM-125, pp. 1-7, November, 1978.

[Sher] J.R. Sherwood, "Data Security in Packet Switched Networks", pp.375-379.

[Stal88] W. Stallings, "Data and Computer Communications", Macmillan Publishing Company, New York, 1988.

[Stal92] W. Stallings, "ISDN and Broadband ISDN", Macmillan Publishing Company, New York, 1992.

[Subb93] W. Subbarao, "ISDN..What is it? Where are we? Where are we going?", North American ISDN Users Forum (NIUF), February, 1994.

[SubWil94] W. Subbarao and M. Williams, "ISDN based Retail Network Development", Annual Review of Communications, April, 94.

[Tanz94] K. Tanzillo, "Gaining ISDN privacy with data encryption", Communications News, pp.54, August, 94.

[Tuch79] W. Tuchman, "Hellman Presents No Shortcut Solutions to the DES", IEEE Spectrum, vol. 16, no. 7, pp. 40-41, July, 1979.

[VoyKen83] V. Voydock and S. Kent, "Security Mechanisms in High-Level Network Protocols", Computing Surveys, vol.15, no. 2, pp.135-171, June, 1983.

[Wild94] S. Wildstrom, "Data Privacy: A Win for Business", BusinessWeek, pp.11, August, 1994.

[Will93] M. Williams, "Development of a Retail Network using ISDN", M.S. Thesis, Florida International University, December, 1993.

APPENDIX 'A'

LIST.C

```
//      LIST.h
//
#ifndef LIST
#define LIST

template <class keyType, class elemType>
class Node {
 public:
  keyType  key;
  elemType elem;
  int height;
  Node<keyType, elemType> *left, *right;
  Node(keyType k, elemType e, Node<keyType, elemType> *l=NULL, Node<keyType, elemType> *r=NULL);
};
//*****************************************
template <class keyType, class elemType>
class List {
 private:
    int height(Node<keyType, elemType> *T);
    int max(int , int);
    Node<keyType,elemType>  *attach(Node<keyType, elemType> *L, Node<keyType, elemType> *R);
    Node<keyType, elemType> *s_rotate_left(Node<keyType, elemType> *T);
    Node<keyType, elemType> *s_rotate_right(Node<keyType, elemType> *T);
    Node<keyType, elemType> *d_rotate_left(Node<keyType, elemType> *T);
    Node<keyType, elemType> *d_rotate_right(Node<keyType, elemType> *T);
 protected:
    Node<keyType, elemType> *tree;

 public:
   List(): tree(NULL) {};
   Node<keyType, elemType>  *insert(keyType, elemType, Node<keyType, elemType> *);
   Node<keyType, elemType>  *remove(keyType, Node<keyType, elemType> *);
   elemType find(keyType k, Node<keyType, elemType> *p);
   Node<keyType, elemType>  *findptr(keyType k, Node<keyType, elemType> *p) ;

   void print(Node< keyType, elemType> *);
   void printtree(Node< keyType, elemType> *,int);

   ~List();
};

#endif
```

I.C

```
//              L.C
//
#include <iostream.h>
#include "List.h"


//*****************************************************************************
//-------------------------------------------------------
//                  private stuff
//-------------------------------------------------------
template <class keyType, class elemType>
int List<keyType, elemType>::height(Node<keyType,elemType> *T){

        if (T==NULL)
                return(0);
        else    return T->height;
};
//-------------------------------------------------------
template <class keyType, class elemType>
int List<keyType, elemType>::max(int x, int y)
{
        if (x>y)
           return(x);
     else
        return(y);
};
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::s_rotate_left(Node<keyType, elemType> *T)
{
        Node<keyType, elemType> *tmp;
     tmp = T->left;
        T->left = tmp->right;
     tmp->right = T;

        T->height  = max(height(T->left), height(T->right)) + 1;
        tmp->height = max(height(tmp->left), T->height) + 1;

        return tmp;
};
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::s_rotate_right(Node<keyType, elemType> *T)
{
        Node<keyType, elemType> *tmp;
     tmp = T->right;
        T->right = tmp->left;
     tmp->left = T;

        T->height  = max(height(T->left), height(T->right)) + 1;
        tmp->height = max(height(tmp->right), T->height) + 1;
```

```
        return tmp;
};
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::d_rotate_left(Node<keyType, elemType> *T)
{
        T->left = s_rotate_right(T->left);
        return(s_rotate_left(T));
};
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::d_rotate_right(Node<keyType, elemType> *T)
{
        T->right = s_rotate_left(T->right );
        return(s_rotate_right(T));
};
//-------------------------------------------------------
//                attach(left,right)
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::attach(Node<keyType, elemType> *L,
Node<keyType, elemType> *R)
{
  if (R != NULL)
  {
    R->left = attach(L,R->left);

    if (height(R->left) - height(R->right) == 2)
    {
      if ( R->key < R->left->key )
        R = s_rotate_left(R);
      else
        R = d_rotate_left(R);
     }
     else
       R->height = max(height(R->left), height(R->right)) + 1;
  }
  else return(L);

  return(R);
};
//-------------------------------------------------------
//                insert(keyType)
//-------------------------------------------------------
template <class keyType, class elemType>
Node<keyType,elemType> *List<keyType, elemType>::insert(keyType k, elemType s, Node<keyType,
elemType> *p) {

    if (p == NULL)
          p = new Node<keyType, elemType>(k, s);
    else if (p->key == k)
      cout<<"in List:insert: the key "<<p->key<<"already exist\n";
    else if (k < p->key)
    {
```

```
        p->left=insert(k, s , p->left);
        if (height(p->left) - height(p->right) == 2)
        {
                    if ( k < p->left->key )
              p = s_rotate_left(p);
           else
              p = d_rotate_left(p);
            }
        else
           p->height = max(height(p->left), height(p->right)) + 1;
        }
         else
      {
        p->right=insert(k, s , p->right);
        if (height(p->right) - height(p->left) == 2)
        {
                    if ( k > p->right->key )
              p = s_rotate_right(p);
           else
              p = d_rotate_right(p);
            }
        else
         p->height = max( height(p->left), height(p->right) ) + 1;
         }
         return (p);
};
//-------------------------------------------------------------
//                    find(keyType)
//-------------------------------------------------------------
template <class keyType, class elemType>
elemType List<keyType, elemType>::find(keyType k, Node<keyType, elemType> *p) {

      if ( p == NULL)
              return( (elemType) 0);
      if (p->key == k)
         return(p->elem);
      if (p->key > k)
         return(find(k, p->left));
      else
         return(find(k, p->right));
};
//-------------------------------------------------------------
//                    findptr(keyType)
//-------------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType> *List<keyType, elemType>::findptr(keyType k, Node<keyType, elemType> *p) {

      if ( p == NULL)
              return(NULL);
      if (p->key == k)
         return(p);
      if (p->key > k)
         return(findptr(k, p->left));
      else
```

```
        return(findptr(k, p->right));
};
//-------------------------------------------------------------
//                  remove(keyType)
//-------------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType>  *List<keyType, elemType>::remove(keyType k, Node<keyType, elemType> *p) {

        Node<keyType, elemType>  *tmp, *tmpl, *tmpr;

    if ( p != NULL)
    {
      if (p->key > k)
        p->left = remove(k, p->left);
      else if (p->key< k)
        p->right = remove(k, p->right);
      else
      {
              p->right = attach(p->left, p->right);
        tmpr = p->right;
        delete p;
        return(tmpr);
      };

      if ( height(p->left) - height(p->right) == 2)
        {  if ( height(p->left->left) < height(p->left->right))
          p = s_rotate_left(p);
        else
          p = d_rotate_left(p);
      }
      else
      {  if (height(p->right) - height(p->left) == 2)
            {  if (height(p->right->right) > height(p->right->left) )
            p = s_rotate_right(p);
          else
            p = d_rotate_right(p);
        }
        else
         p->height = max( height(p->left), height(p->right) ) + 1;
      };
    };
        return (p);
};
//-------------------------------------------------------------
//                  ~List()
//-------------------------------------------------------------
template <class keyType, class elemType>
List<keyType, elemType>::~List() {
        Node<keyType,elemType> *tmp=tree, *currunt;
}

//-------------------------------------------------------------
//                  List::print()
//-------------------------------------------------------------
```

```
template <class keyType, class elemType>
void List< keyType, elemType>::print(Node<keyType, elemType> *p) {

      if (p != NULL){
         print( p->left);
         cout<< p->key<<"   "<<p->elem<< endl;
         print( p->right);
      }
}
//-------------------------------------------------------------
//                  List::print tree()
//-------------------------------------------------------------
template <class keyType, class elemType>
void List< keyType, elemType>::printtree(Node< keyType, elemType> *p, int i){

      if (p != NULL){
         printtree( p->right,i+5);
         for(int j=0; j<i; j++)
            cout<<" ";
         cout<<p->key <<endl;
         printtree( p->left,i+5);
      }
}
//-------------------------------------------------------------
//                Node::Node(keyType, elemType)
//-------------------------------------------------------------
template <class keyType, class elemType>
Node<keyType, elemType>::Node(keyType k, elemType s, Node *l=NULL, Node *r=NULL){
          key  = k;
          elem  = s;
          left  = l;
          right = r;
     height = 1;
}
```

doctor.C

```
//                  doctor.C
//
// DOCTOR MODULE

#include <signal.h>
#include <string.h>
#include <String.h>
#include <fstream.h>
#include <stdio.h>
#include <fcntl.h>
#include "itoa.h"
#define ESC 27

char doctors_name[80];
char doctors_display[80];
char *server;

void patien(void);
void lab(void);
void pharmacy(void);


struct rec {
        char name[30];
        char lname[30];
            char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      };

char *datafile = "datafile";
char *ptrfile = "ptrfile";

void  close_up_exit();


int catchint(int signum)
{
     cout<<"\n*****************************\n";
     cout<<"signal -----"<<signum;
     cout<<"\n*****************************\n";
     close_up_exit();
     return(0);
};
//---------------------------------------------------------------
class Client{
  private:
       char choice[5];
       int pdr,rp;    //pdr: first pip to read the server, rp main pip to work with the server
       int pdw,wp; //pdr  first pip to write to server, wp main pip to work with the server
       char *rdpip, *wrpip;
       int tmpr,tmp;
```

```
        char key[20];
        char recname[20] ;
        char element[20];

    public:
        Client();
        int add();
        void add_new_record();
        int delet();
        int find();
        int Display_options();
        void print();
        void whatever();
        void quit();
        ~Client();
};
//---------------------------------------------------
Client::Client()
{
   int pid;
   char tempbuf[40];
   char *newpip;
   server = new char[8];

   tempbuf[0]='\0';
   if ( (pdw = open("pip1",O_WRONLY)) <0){
       cout<<"\n\n debug2..error opening the pip1  \n\n";
       exit(1);
   };
   pid=getpid();
   newpip=itoa(pid);
   rdpip=itoa(pid);
   wrpip=itoa(pid);
   strcat(rdpip,"a");
   strcat(wrpip,"b");
   //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   if(mknod(rdpip,010666,0)<0){
     cout<<"\nmknod  "<< rdpip<<"\n";
   };
   if(mknod(wrpip,010666,0)<0){
     cout<<"\nmknod  "<<wrpip<<" \n";
   };
   write(pdw,newpip,8);

   if ((tmp = open(wrpip,O_RDONLY|O_NDELAY))<0){
     cout<<"\nopenning "<<rdpip<<" \n\n";
     exit(1);
   };
   if ((wp = open(wrpip,O_WRONLY|O_NDELAY))<0){
     cout<<"\n\nopening the "<<wrpip<<" \n\n";
     exit(1);
   };
   if ((rp = open(rdpip,O_RDONLY|O_NDELAY))<0){
     cout<<"\n\n openning "<<rdpip<<" \n\n";
```

```
      exit(1);
    };
    //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

};
//-------------------------------------------------
Client::~Client()
{
    close(pdr);
    close(tmp);
    close(rp);
    close(wp);
};
//-------------------------------------------------
int Client::add()
{
    cout<<"please enter the record name: " ;
    cin >> recname;
    write(wp,choice,5);
    write(wp,recname,20);
    return 0;
};
//-------------------------------------------------
int Client::delet()
{
    cout<<"please enter the record name: " ;
    cin >> recname;
    write(wp,choice,5);
    write(wp,recname,20);
    return 0;
};
//-------------------------------------------------
int Client::find()
{
    char answr;
    char elem[20];
    char recname[80];
    char patient[80];
    char appendrec[80];
    char patientdisplay[80];
    char pdisplay[2][50];
    char doctorname[80];
    char doctordisplay[80];
    char ddisplay[2][50];
    String key1;
    String key2;
    String key3;
    char displaypic[80];
    char displayrec[80];
    char *k;
    struct rec *r;
    long offset,position;
    int i,n;
    i=0;
```

```
    n=0;
    r=new rec;

    cout<<"\nenter  patient's name: ";

    cin >> recname;
    strcpy(patient,recname);

    write(wp,choice,5);
    write(wp,recname,20);
    while ( read(rp,elem,20) <0)
      ;
    strcpy(recname,elem);
    strcat(recname,".rec");
    ifstream inFile (recname,ios::in);
    inFile >> pdisplay[0];
    inFile >> pdisplay[1];
    inFile.close();

    cout<<"patient display is "<< pdisplay[1]<<" (y/n):";
    cin>>answr;
    if (answr=='n')
    {
       cout<<"please enter the patient's display name: ";
       cin >>pdisplay[1];
    }


 // at patient display

    strcpy(patientdisplay,"xv -display ");
    strcat(patientdisplay, pdisplay[1] );
    strcat(patientdisplay, ":0.0 ");
    strcat(patientdisplay,"-name  DR_");
    strcat(patientdisplay,doctors_name);
    strcat(patientdisplay,"  ");
    strcat(patientdisplay,doctors_name);
    strcat(patientdisplay,".gif& ");
    system(patientdisplay);

 // at doctors display  picture  of patient

    strcpy(doctordisplay,"xv -display  ");
    strcat(doctordisplay,doctors_display);
    strcat(doctordisplay,":0.0 ");

    strcat(doctordisplay,"-name  ");
    strcat(doctordisplay,patient);
    strcat(doctordisplay," ");

    strcat(doctordisplay,elem);
    strcat(doctordisplay,".gif");
    strcat(doctordisplay,"& ");
    system(doctordisplay);
```

```
// at doctors display  file of the   patient

    strcpy(displayrec,"emacs -display  ");
    strcat(displayrec ,doctors_display);
    strcat(displayrec ,":0.0 ");
    strcat(displayrec,elem);
    strcat(displayrec,".rec");
    strcat(displayrec,"& ");
    cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";
    system(displayrec);
    cout<<"\nprescription form (y/n): ";
    cin >> answr;
    if (answr=='y')
    {
       strcpy(displayrec,"emacs -display  ");
       strcat(displayrec ,doctors_display);
            strcat(displayrec ,":0.0 ");
            strcat(displayrec,elem);
            strcat(displayrec,".phar");
       cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";
       system(displayrec);

       // display prescription at patient display
       strcpy(displayrec,"xv -display  ");
       strcat(displayrec ,pdisplay[1]);
       strcat(displayrec ,":0.0 ");
       strcat(displayrec,elem);
       strcat(displayrec,".phar& ");
       cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";

       // display prescription at  phamacy  display
       strcpy(appendrec,"cat ");
       strcat(appendrec,elem);
       strcat(appendrec,".phar");
       strcat(appendrec," >> ");
       strcat(appendrec,elem);
       strcat(appendrec,".rec ");
       system(appendrec);
    }
    cout<<"send to pharmacy printer(y/n)? ";
    cin>>answr;
    if (answr=='y' )
    {     strcpy(appendrec,"lpr  -Pisdn ");
          strcat(appendrec,elem);
          strcat(appendrec,".phar&");
          system(appendrec);
    }
    return 0;
};
//--------------------------------------------------
void Client::print(){
    write(wp,choice,5);
};
```

```
//-------------------------------------------------
void Client::whatever(){
   cout<<"\n\n WHAT EVER.....\n";
};
//-------------------------------------------------
void Client::quit(){
   cout<<"\n\n bye .....\n\n";
   write(wp,choice,5);
};
//-------------------------------------------------
int Client::Display_options(){


   cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
   cout<<"\n WELCOME TO MEDNET\n";
   cout<<"-----------------------\n\n";
   cout<<"\n+++++++++++++++++++++++++++++++\n";
   cout<<"1)  add a new record \n";
   cout<<"2)  find a  record \n";
   cout<<"3)  quit \n";
   cout<<"\n+++++++++++++++++++++++++++++++\n";
   cout<<"\n please enter your choice: ";
   cin >> choice;
   return (atoi(choice));
}
//-------------------------------------------------
void  close_up_exit()
{
   int  serverpid;
   cout<<"cln  "<<getpid()<<"  is killed"<< endl;
   cout<<"\n==== finished====\n";

   serverpid=atoi(server);
   cout<<"server pid = "<<serverpid<<endl;
   kill(serverpid,SIGTERM);

   exit(0);

}
//-------------------------------------------------------------------
void Client::add_new_record()
{
      struct rec  *r;
      char ans[30],record[30],filename[30];
      long position;
      r=new rec;
      char append[80];
          FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

      while(1)
      {
        do{
           cout <<"Add a new record(yes/no): "; cin>> ans;
           if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0 | \
```

```
            (strcmp(ans,"YES"))==0 | (strcmp(ans,"yes"))==0)
                break;
            else cout<<"\n\n\aAnswer enter yes or no\a\n\n";
        }while(1);
        if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0)
            break;
        cout<<"Soc Sec. No. : "; cin >> r->ssn;
        cout<<"Last  Name---: "; cin >> r->lname;
        filename[0]='\0';
        strcat(filename,"cp templatefile ");
        strcat(filename,r->ssn);
        strcat(filename,".rec");
        system(filename);
        filename[0]='\0';

        strcat(filename,"cp templatephar  ");
        strcat(filename,r->ssn);
        strcat(filename,".phar");
        system(filename);
        record[0]='\0';

        strcpy(record,"emacs -display ");
        strcat(record,doctors_display);
        strcat(record,":0.0 ");
        strcat(record,r->ssn);
        strcat(record,".rec");
        strcat(record,"& ");
        cout<< "record = "<<record<<endl;
        system(record);

        pfp=fopen(ptrfile,"a+");
        fprintf(pfp,"%s  %s\n",r->lname,r->ssn);
        fclose(pfp);

        write(wp,choice,5);
        write(wp,r->lname,20);
        write(wp,r->ssn,20);
      }
}

//-------------------------------------------------
int mainmenu()
{
   int select;
   system("cls");
   cout<<"main menu\n\n\n";
   cout<<"1) patient records\n";
   cout<<"2) pharmacy\n";
   cout<<"3) medical lab\n";
   cout<<"4) quit\n";
   cout<<"------------------------\n";
   cout<<"please make your selection: ";
   do {
      cin>> select;
```

```
        if (select !=1 && select !=2 && select !=3 && select !=4 )
            cout<<"\n\ainvalid selection\n"<<"please make your selection again: ";
    } while( select !=1 && select !=2 && select !=3 && select !=4 ) ;

    return select;

}
//---------------------------------------------------------------

void patient()
{
    Client cln;
    int choice;
    while(1)
    {
        choice = cln.Display_options();
        switch(choice)
        {
           case 1:  cln.add_new_record();
                    break;
           case 2:  cln.find();
                    break;
           case 3:  return;
           case 4:  cln.print();
                    break;
           case 5:  cln.delet();
                    break;
           default: break;
         }
    }
}

//*********************************************************************
main()
{

      char answr;
      signal(SIGINT,  catchint);
      signal(SIGHUP,  catchint);
      signal(SIGBUS,  catchint);
      signal(SIGTERM, catchint);
      system("cls");
      cout<<"\n WELCOME TO MEDNET\n";
      cout<<"-----------------------\n\n";
      cout<<"please enter doctors name: ";
      cin>>doctors_name;
      cout<<"\nPlease enter doctors display name: ";
      cin>> doctors_display;
      patient();
}
```

phar.C

```
//                    phar.C
//
// PHARMACY   MODULE

#include <signal.h>
#include <string.h>
#include <String.h>
#include <fstream.h>
#include <stdio.h>
#include <fcntl.h>
#include "itoa.h"
#define ESC 27

char phar_name[80];
char phar_display[80];

char *server;

void patien(void);
void lab(void);
void pharmacy(void);


struct rec {
        char name[30];
        char lname[30];
            char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      };

char *datafile = "datafile";
char *ptrfile = "pharptrfile";

void  close_up_exit();


int catchint(int signum)
{
     cout<<"\n*****************************\n";
     system("cls");
     cout<<"bye\n";
     close_up_exit();
     exit(0);
};
//---------------------------------------------------------------
class Client{
  private:
       char choice[5];
       int pdr,rp;    //pdr: first pip to read the server, rp main pip to work with the server
       int pdw,wp; //pdr  first pip to write to server, wp main pip to work with the server
       char *rdpip, *wrpip;
```

```
        int tmpr,tmp;
        char key[20];
        char recname[20] ;
        char element[20];

    public:
        Client();
        int add();
        void add_new_record();
        int delet();
        int find();
        int Display_options();
        void print();
        void whatever();
        void quit();
       ~Client();
};
//---------------------------------------------------
Client::Client()
{
  int pid;
  char tempbuf[40];
  char *newpip;
  server = new char[8];

  tempbuf[0]='\0';
  if ( (pdw = open("ppip1",O_WRONLY)) <0){
      cout<<"\n----------\n";
      exit(1);
  };
      pid=getpid();
      newpip=itoa(pid);
      rdpip=itoa(pid);
      wrpip=itoa(pid);
      strcat(rdpip,"a");
      strcat(wrpip,"b");
      //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      if(mknod(rdpip,010666,0)<0){
        cout<<"\n----------\n";
      };
      if(mknod(wrpip,010666,0)<0){
        cout<<"\n---------- \n";
      };
      write(pdw,newpip,8);

      if ((tmp = open(wrpip,O_RDONLY|O_NDELAY))<0){
        cout<<"\n----------\n";
        exit(1);
      };
       if ((wp = open(wrpip,O_WRONLY|O_NDELAY))<0){
        cout<<"\n----------\n";
        exit(1);
      };
       if ((rp = open(rdpip,O_RDONLY|O_NDELAY))<0){
```

```
        cout<<"\n----------\n";
        exit(1);
      };

      //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

};
//---------------------------------------------------
Client::~Client()
{
   close(pdr);
   close(tmp);
   close(rp);
   close(wp);
};
//---------------------------------------------------
int Client::add()
{
   cout<<"please enter the record name: " ;
   cin >> recname;
   write(wp,choice,5);
   write(wp,recname,20);
   return 0;
};
//---------------------------------------------------
int Client::delet()
{
   cout<<"please enter the record name: " ;
   cin >> recname;
   write(wp,choice,5);
   write(wp,recname,20);
   return 0;
 };
//---------------------------------------------------
int Client::find()
{
   char answr;
   char elem[20];
   char recname[80];
   char patient[80];
   char docname[80];
   char Doctorname[2][80];
   char patientdisplay[80];
   char docdisplay[80];
   char pdisplay[2][50];
   char doctorname[80];
   char doctordisplay[80];
   char labdisplay[80];
   char ddisplay[2][50];
   String key1;
   String key2;
   String key3;
   char displaypic[80];
   char displayrec[80];
```

```
    char *k;
    struct rec *r;
    long offset,position;
    int i,n;
    i=0;
    n=0;
    r=new rec;

    cout<<"\nenter  the patient's name: ";

    cin >> recname;
    strcpy(patient,recname);

    write(wp,choice,5);
    write(wp,recname,20);
    while ( read(rp,elem,20) <0)
      ;
    strcpy(recname,elem);
    strcat(recname,".phar");
    ifstream inFile (recname,ios::in);

    inFile.close();

    cout<<"please enter the doctors 's name: ";
    cin >> docname ;
    cout<<"please enter the doctor's display name: ";
    cin >>ddisplay[1];


// at doctors  display  picture

    strcpy(docdisplay,"xv -display ");
    strcat(docdisplay, ddisplay[1] );
    strcat(docdisplay, ":0.0  ");
    strcat(docdisplay,"-name  Phar_");
    strcat(docdisplay,phar_name);
    strcat(docdisplay,"  ");
    strcat(docdisplay,phar_name);
    strcat(docdisplay,".gif& ");
    cout<<"docdisplay =  "<<docdisplay<<endl;
    system(docdisplay);

// at pharmacy  display  patient picture

    strcpy(labdisplay,"xv -display  ");
    strcat(labdisplay,phar_display);
    strcat(labdisplay,":0.0 ");

    strcat(labdisplay,"-name ");
    strcat(labdisplay,patient);
    strcat(labdisplay," ");

    strcat(labdisplay,elem);
    strcat(labdisplay,".gif");
```

```
    strcat(labdisplay,"& ");
    cout<<"\nlabdisplay"<<labdisplay <<"\n";
    system(labdisplay);

// at pharmacy  display  doctors  picture

    strcpy(labdisplay,"xv -display  ");
    strcat(labdisplay,phar_display);
    strcat(labdisplay,":0.0 ");

    strcat(labdisplay,"-name  DR_");
    strcat(labdisplay,docname );
    strcat(labdisplay," ");

    strcat(labdisplay,docname );
    strcat(labdisplay,".gif");
    strcat(labdisplay,"& ");
    cout<<"\nlabdisplay"<<labdisplay <<"\n";
    system(labdisplay);


//  display the record

    strcpy(displayrec,"xv -display  ");
    strcat(displayrec ,phar_display);
    strcat(displayrec ,":0.0 ");
    strcat(displayrec,elem);
    strcat(displayrec,".phar");
    strcat(displayrec,"& ");
    cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";
    system(displayrec);

    return 0;
};
//-------------------------------------------------
void Client::print(){
    write(wp,choice,5);
};
//-------------------------------------------------
void Client::whatever(){
    cout<<"\n\n WHAT EVER.....\n";
};
//-------------------------------------------------
void Client::quit(){
    cout<<"\n\n bye .....\n\n";
    write(wp,choice,5);
};
//-------------------------------------------------
int Client::Display_options(){


    cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
    cout<<"\n WELCOME TO MEDNET\n";
    cout<<"----------------------\n\n";
```

```
    cout<<"\n+++++++++++++++++++++++++++++++\n";
    cout<<"1)  add a new record \n";
    cout<<"2)  find a  record \n";
    cout<<"3)  quit \n";
    cout<<"\n+++++++++++++++++++++++++++++++\n";
    cout<<"\n please enter your choice: ";
    cin >> choice;
    return (atoi(choice));
}
//-------------------------------------------------
void  close_up_exit()
{
   int  serverpid;
   cout<<"cln  "<<getpid()<<"  is killed"<< endl;
   cout<<"\n==== finished====\n";

   serverpid=atoi(server);
   cout<<"server pid = "<<serverpid<<endl;
   kill(serverpid,SIGTERM);

   exit(0);

}
//-------------------------------------------------------------------
void Client::add_new_record()
{
      struct rec  *r;
      char ans[30],record[30],filename[30];
      long position;
      r=new rec;
      char append[80];
          FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

      while(1)
      {
        do{
           cout <<"Add a new record(yes/no): "; cin>> ans;
           if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0 | \
               (strcmp(ans,"YES"))==0 | (strcmp(ans,"yes"))==0)
                   break;
             else cout<<"\n\n\aAnswer enter yes or no\a\n\n";
        }while(1);
        if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0)
            break;
        cout<<"Soc Sec. No. : "; cin >> r->ssn;
        cout<<"Last  Name---: "; cin >> r->lname;
        record[0]='\0';

        pfp=fopen(ptrfile,"a+");
        fprintf(pfp,"%s  %s\n",r->lname,r->ssn);
        fclose(pfp);

        write(wp,choice,5);
        write(wp,r->lname,20);
```

```
            write(wp,r->ssn,20);
        }
}

//-----------------------------------------------
int mainmenu()
{
   int select;
   system("cls");
   cout<<"main menu\n\n\n";
   cout<<"1) patient records\n";
   cout<<"2) pharmacy\n";
   cout<<"3) medical lab\n";
   cout<<"4) quit\n";
   cout<<"-----------------------\n";
   cout<<"please make your selection: ";
   cin>> select;
   return select;

}
//-----------------------------------------------------------------

void patient()
{
   Client cln;
   int choice;
   while(1)
   {
        choice = cln.Display_options();
        switch(choice)
        {
           case 1:  cln.add_new_record();
                    break;
           case 2:  cln.find();
                    break;
           case 3:  return;
           case 4:  cln.print();
                    break;
           default: break;
        }
   }
}
//*********************************************************************
main()
{
      char answr;
      signal(SIGINT,  catchint);
      signal(SIGHUP,  catchint);
      signal(SIGBUS,  catchint);
      signal(SIGTERM, catchint);
      system("cls");
      cout<<"\n WELCOME TO MEDNET PHARMACY \n";
      cout<<"-----------------------\n\n";
      cout<<"please enter pharmacist's  name: ";
```

```
    cin>>phar_name;
    cout<<"\nPlease enter pharmacy's display name: ";
    cin>> phar_display;
    patient();
}
```

lab.C

```
//                    lab.C
//
// LAB   MODULE

#include <signal.h>
#include <string.h>
#include <String.h>
#include <fstream.h>
#include <stdio.h>
#include <fcntl.h>
#include "itoa.h"
#define ESC 27

char labtech_name[80];
char labtech_display[80];

char *server;

void patien(void);
void lab(void);
void pharmacy(void);


struct rec {
        char name[30];
        char lname[30];
            char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      } ;

char *datafile = "datafile";
char *ptrfile = "labptrfile";

void  close_up_exit();


int catchint(int signum)
{
    cout<<"\n*****************************\n";
    //cout<<"signal -----"<<signum;
    system("cls");
    cout<<"bye\n";
    close_up_exit();
    exit(0);
};
//---------------------------------------------------------------
class Client{
  private:
      char choice[5];
      int pdr,rp;    //pdr: first pip to read the server, rp main pip to work with the server
      int pdw,wp; //pdr  first pip to write to server, wp main pip to work with the server
```

```
        char *rdpip, *wrpip;
        int tmpr,tmp;
        char key[20];
        char recname[20] ;
        char element[20];

    public:
        Client();
        int add();
        void add_new_record();
        int delet();
        int find();
        int Display_options();
        void print();
        void whatever();
        void quit();
       ~Client();
};
//---------------------------------------------------
Client::Client()
{
  int pid;
  char tempbuf[40];
  char *newpip;
  server = new char[8];

  tempbuf[0]='\0';
  if ( (pdw = open("lpip1",O_WRONLY)) <0){
      cout<<"\n\n debug2..error opening the pip1  \n\n";
      exit(1);
  };
      pid=getpid();
      newpip=itoa(pid);
      rdpip=itoa(pid);
      wrpip=itoa(pid);
      strcat(rdpip,"a");
      strcat(wrpip,"b");
     //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      if(mknod(rdpip,010666,0)<0){
        cout<<"\nerror mknod  "<< rdpip<<"\n";
      };
      if(mknod(wrpip,010666,0)<0){
        cout<<"\nerror mknod  "<<wrpip<<" \n";
      };
      write(pdw,newpip,8);

      if ((tmp = open(wrpip,O_RDONLY|O_NDELAY))<0){
        cout<<"\n\nError openning "<<rdpip<<" \n\n";
        exit(1);
      };
      if ((wp = open(wrpip,O_WRONLY|O_NDELAY))<0){
        cout<<"\n\n error opening the "<<wrpip<<" \n\n";
        exit(1);
      };
```

```
      if ((rp = open(rdpip,O_RDONLY|O_NDELAY))<0){
        cout<<"\n\nError openning "<<rdpip<<" \n\n";
        exit(1);
      };

      //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

};
//---------------------------------------------------
Client::~Client()
{
   close(pdr);
   close(tmp);
   close(rp);
   close(wp);
};
//---------------------------------------------------
int Client::add()
{
   cout<<"please enter the record name: " ;
   cin >> recname;
   write(wp,choice,5);
   write(wp,recname,20);
   return 0;
};
//---------------------------------------------------
int Client::delet()
{
   cout<<"please enter the record name: " ;
   cin >> recname;
   write(wp,choice,5);
   write(wp,recname,20);
   return 0;
 };
//---------------------------------------------------
int Client::find()
{
   char answr;
   char elem[20];
   char recname[80];
   char patient[80];
   char docname[80];
   char patientdisplay[80];
   char docdisplay[80];
   char pdisplay[2][50];
   char doctorname[80];
   char doctordisplay[80];
   char labdisplay[80];
   char ddisplay[2][50];
   String key1;
   String key2;
   String key3;
   char displaypic[80];
   char displayrec[80];
```

```
    char *k;
    struct rec *r;
    long offset,position;
    int i,n;
    i=0;
    n=0;
    r=new rec;

    cout<<"\nenter patient's name: ";

    cin >> recname;
    strcpy(patient,recname);

    write(wp,choice,5);
    write(wp,recname,20);
    while ( read(rp,elem,20) <0)
      ;
    strcpy(recname,elem);
    strcat(recname,".lab");
    ifstream inFile (recname,ios::in);
    inFile >> ddisplay[0];
    inFile >> ddisplay[1];
    inFile.close();

    cout<<"please enter the doctor's name: ";
    cin >> docname ;

    cout<<"please enter the doctor's display name: ";
    cin >>ddisplay[1];

    strcpy(docdisplay,"xv -display ");
    strcat(docdisplay, ddisplay[1] );
    strcat(docdisplay, ":0.0  ");
    strcat(docdisplay,"-name Labtech_");
    strcat(docdisplay,labtech_name);
    strcat(docdisplay,"  ");
    strcat(docdisplay,labtech_name);
    strcat(docdisplay,".gif& ");
    cout<<"docdisplay =  "<<docdisplay<<endl;
    system(docdisplay);

// at lab  display  patient picture

    strcpy(labdisplay,"xv -display  ");
    strcat(labdisplay,labtech_display);
    strcat(labdisplay,":0.0 ");

    strcat(labdisplay,"-name ");
    strcat(labdisplay,patient);
    strcat(labdisplay," ");

    strcat(labdisplay,elem);
    strcat(labdisplay,".gif");
    strcat(labdisplay,"& ");
```

```
    cout<<"\nlabdisplay"<<labdisplay <<"\n";
    system(labdisplay);

 // at lab  display  doctors  picture

    strcpy(labdisplay,"xv -display  ");
    strcat(labdisplay,labtech_display);
    strcat(labdisplay,":0.0 ");

    strcat(labdisplay,"-name  DR_");
    strcat(labdisplay,docname);
    strcat(labdisplay," ");

    strcat(labdisplay,docname);
    strcat(labdisplay,".gif");
    strcat(labdisplay,"& ");
    cout<<"\nlabdisplay"<<labdisplay <<"\n";
    system(labdisplay);


 //  display the record

    strcpy(displayrec,"emacs -display  ");
    strcat(displayrec ,labtech_display);
    strcat(displayrec ,":0.0 ");
    strcat(displayrec,elem);
    strcat(displayrec,".lab");
    strcat(displayrec,"& ");
    cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";
    system(displayrec);

    strcpy(displayrec,"xv -display  ");
    strcat(displayrec,ddisplay[1]);
    strcat(displayrec ,":0.0 ");
    strcat(displayrec,elem);
    strcat(displayrec,".lab");
    strcat(displayrec,"& ");
    cout<<"\n++++++\n"<<displayrec<<"\n++++++++\n";
    system(displayrec);

    strcpy(displayrec,"cat   ");
    strcat(displayrec,elem);
    strcat(displayrec,".lab");
    strcat(displayrec," >> ");
    strcat(displayrec,elem);
    strcat(displayrec,".rec");
    system(displayrec);


    return 0;
};
//-------------------------------------------------
void Client::print(){
    write(wp,choice,5);
```

```
};
//-------------------------------------------------
void Client::whatever(){
   cout<<"\n\n WHAT EVER.....\n";
};
//-------------------------------------------------
void Client::quit(){
   cout<<"\n\n bye .....\n\n";
   write(wp,choice,5);
};
//-------------------------------------------------
int Client::Display_options(){


   cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
   cout<<"\n WELCOME TO MEDNET\n";
   cout<<"-----------------------\n\n";
   cout<<"\n++++++++++++++++++++++++++++++++\n";
   cout<<"1)  add a new record \n";
   cout<<"2)  find a  record \n";
   cout<<"3)  quit \n";
   cout<<"\n++++++++++++++++++++++++++++++++\n";
   cout<<"\n please enter your choice: ";
   cin >> choice;
   return (atoi(choice));
}
//-------------------------------------------------
void  close_up_exit()
{
   int  serverpid;
   cout<<"cln  "<<getpid()<<"  is killed"<< endl;
   cout<<"\n==== finished====\n";

   serverpid=atoi(server);
   cout<<"server pid = "<<serverpid<<endl;
   kill(serverpid,SIGTERM);

   exit(0);

}
//-----------------------------------------------------------------
void Client::add_new_record()
{
      struct rec  *r;
      char ans[30],record[30],filename[30];
      long position;
      r=new rec;
      char append[80];
          FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

      while(1)
      {
        do{
           cout <<"Add a new record(yes/no): "; cin>> ans;
```

```
            if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0 | \
                (strcmp(ans,"YES"))==0 | (strcmp(ans,"yes"))==0)
                    break;
                else cout<<"\n\n\aAnswer enter yes or no\a\n\n";
            }while(1);
            if((strcmp(ans,"NO")) ==0 | (strcmp(ans,"no")) ==0)
                break;
            cout<<"Soc Sec. No. : "; cin >> r->ssn;
            cout<<"Last  Name---: "; cin >> r->lname;
            filename[0]='\0';
            strcat(filename,"cp templatelab ");
            strcat(filename,r->ssn);
            strcat(filename,".lab");
            system(filename);
            record[0]='\0';

            strcpy(record,"emacs -display ");
            strcat(record,labtech_display);
            strcat(record,":0.0 ");
            strcat(record,r->ssn);
            strcat(record,".lab");
            strcat(record,"& ");
            cout<< "record = "<<record<<endl;
            system(record);

            pfp=fopen(ptrfile,"a+");
            fprintf(pfp,"%s  %s\n",r->lname,r->ssn);
            fclose(pfp);

            write(wp,choice,5);
            write(wp,r->lname,20);
            write(wp,r->ssn,20);
        }
}

//-------------------------------------------------------
int mainmenu()
{
   int select;
   system("cls");
   cout<<"main menu\n\n\n";
   cout<<"1) patient records\n";
   cout<<"2) pharmacy\n";
   cout<<"3) medical lab\n";
   cout<<"4) quit\n";
   cout<<"-----------------------\n";
   cout<<"please make your selection: ";
   cin>> select;
   return select;

}
//-------------------------------------------------------------

void patient()
```

```
{
    Client cln;
    int choice;
    while(1)
    {
        choice = cln.Display_options();
        switch(choice)
        {
           case 1:  cln.add_new_record();
                    break;
           case 2:  cln.find();
                    break;
           case 3:  return;
           case 4:  cln.print();
                    break;
           default: break;
        }
    }
}


//*************************************************************************
main()
{

//char doctors_name[80];
//char doctors_display[80];

      char answr;
      signal(SIGINT,  catchint);
      signal(SIGHUP,  catchint);
      signal(SIGBUS,  catchint);
      signal(SIGTERM, catchint);
      system("cls");
      cout<<"\n WELL COME TO MEDNET\n";
      cout<<"-----------------------\n\n";
      cout<<"please enter lab tech. name: ";
      cin>>labtech_name;
      cout<<"\nPlease enter lab display name: ";
      cin>> labtech_display;
      patient();
}
```

dserver.C

```
//              dserver.C


#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>
#include "L.C"
#include "itoa.h"
#define msgsize 80
struct rec {
        char name[30];
        char lname[30];
        char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      } ;

int pdr;
int pdw;
int rp;
int wp;
int pid[2];
char inbuf[msgsize];
```

```
//                    pserver.C

#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>
#include "L.C"
#include "itoa.h"
#define msgsize 80
struct rec {
        char name[30];
        char lname[30];
        char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      } ;

int pdr;
int pdw;
int rp;
int wp;
int pid[2];
char inbuf[msgsize];
char *datafile = "ldatafile";
char *ptrfile = "pharptrfile";
char *server;

char clnpid[80],rdpip[80],wrpip[80];
int tmp;
void  close_up_exit();

//*********************************************
int catchint(int signum)
{    int clnpid;
     cout<<"\n*****************************\n";
     cout<<"signal -----"<<signum;
     cout<<"\n*****************************\n";
     close_up_exit();
     exit(0);
};

//*********************************************
template <class keyType, class elemType>

class Dict: public List<keyType, elemType>{
  private:
     char *fname;
  public:
     Dict(){};
```

```
    Dict(char *);
    int  add(keyType k, elemType);
    elemType find(keyType k);
    void delet(keyType);
    void prnt();
    void lookup(){};
    void find(){};

    void operator[](){};
    ~Dict(){};
};


//---------------------------------
//      Dict(filename)
//---------------------------------
template <class keyType, class elemType>
Dict<keyType, elemType>::Dict(char *ptrfile)
{
  keyType key ;
  elemType elem ;

  ifstream inFile (ptrfile,ios::in);
  cout<<"ptrfile="<<ptrfile<<endl;
  while(!inFile.eof()){
     inFile >> key;
     inFile >> elem;
     add(key,elem);
  }
  inFile.close();

};


//---------------------------------
//      add(key,elem)
//---------------------------------
template <class keyType, class elemType>
int Dict<keyType, elemType>::add(keyType k, keyType e){

        tree = insert(k,e,tree);
        return 1;
};
//-----------------------------------
//display picture
//-----------------------------------

void disply(char elem[])
{

  int n=0;
  char display[80];
  char *k;
  struct rec *r;
```

```
    long offset,position;
    r=new rec;
    FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

    offset = atoi(elem);
    if(offset == 0)
       cout<<"\n\n---------\a\a record not found\n\n";
    else{
       dfp=fopen(datafile,"rb");
       fseek(dfp,offset,0);
       fread((char *)  r, sizeof(struct rec),1,dfp);
       fclose(dfp);
       strcpy(display,"xv ");
       strcat(display,r->name);
       strcat(display," &");
       cout<<"\n++++++\n"<<display<<"\n++++++++\n";
       system(display);
    }
};

//---------------------------------
//       find(key)
//---------------------------------
template <class keyType, class elemType>
elemType Dict<keyType, elemType>::find(keyType k){

      char *element = List<keyType, elemType>::find(k,tree);
      cout<<"\n before if.....\n";
      if (element != 0){
         cout<<"the element of key "<<k<<" is "<<element<<endl;
         //============================
         //display(element);
         //============================
         return element;
      }
      else{
         cout<<"key "<< k <<" does not exist \n";
         return ("notfound");
      };
};
//---------------------------------
//       delet(key)
//---------------------------------
template <class keyType, class elemType>
void Dict<keyType, elemType>::delet(keyType key){

      cout<<"removing "<<key<<endl;
          remove(key,tree);
};
//---------------------------------
//       print()
//---------------------------------
template <class keyType, class elemType>
void Dict<keyType, elemType>::prnt(){
```

```
        cout<<"-----------------------------\n";
            cout<<"\nprinting the sorted List\n\n";
        print(tree);
        cout<<"-----------------------------\n";
        printtree(tree,45);
        cout<<"-----------------------------\n";
};
//*************************************************
//-------------------------------------------------
template <class keyType, class elemType>
int console(Dict<keyType, elemType>  *D,int pd){

      int select;
      char choice[5];
      char key[20];
      char elem[20];
      char *element ;
      int quit=0;
      if((read(rp,choice,5))>0)
      {  cout<<choice<<"_________--------_________\n";
        select = atoi(choice);
        sleep(1);
        switch(select)
            {
              case 1 :  read(rp,key,20);
                 read(rp,elem,20);
                 D->add(key,elem);              //add
                            inbuf[0]='\0';
                            sprintf(inbuf,"%s  %s\n",key,elem);
                 write(pid[1],inbuf,msgsize);
                 break;

              case 2 :  read(rp,key,20);
                 element=(char *) D->find(key);   //find
                 write(wp,element,20);
                   break;

              case 4 :  cout<<"parent is printing\n";     //print
                     D->prnt();
                            break;

                 case  5 :  read(rp,key,20);
                 D->delet(key);                 //delete
                            inbuf[0]='\0';
                   sprintf(inbuf,"%s",key);
                 write(pid[1],inbuf,msgsize);
                 break;


              case 6 :  quit = 1;

            default:  break;
        }
```

```
        }
        return quit;
};
//-------------------------------------
void set_up_newpip( char *newpip , char  *rdpip, char *wrpip)
{
        int tmp;
        int quit=0;
        char *pid;

        cout<<"\ndebug2----the new pipe is: "<<newpip<<"\n";
        strcpy(rdpip,newpip);
        strcpy(wrpip,newpip);
        strcat(wrpip,"a");
        strcat(rdpip,"b");
        if ( (wp = open(wrpip,O_WRONLY )) <0){
          cout<<"\n----------\n";
          exit(1);
        };

        if ( (rp = open(rdpip,O_RDONLY) )<0 ){
          cout<<"\n\n----------1 \n";
          exit(1);
        };
        pid= itoa(getpid());
        cout<<"pid  "<<pid<<endl;
        cout<<"wrpip is: "<<wrpip<<endl;
        cout<<"rdpip is : "<<rdpip<<endl;
}

//-------------------------------------
void  close_up_exit()
{
        close(tmp);
        close(pdw);
        close(rp);
        close(wp);
        unlink(rdpip);
        unlink(wrpip);
        cout<<"server  "<<getpid()<<"  is killed"<< endl;
        cout<<"\n==== finished====\n";
        kill(atoi(clnpid),SIGTERM);
        exit(0);
}
//*************************************************
main(){

        int quit=0;

        int INT_signal();
        Dict<String,String>  D(ptrfile);

            char command[5];
            char rec[60];
```

```
        char name[60];
        char ptr[60];
    int  cmd;
        int pd;
    signal(SIGINT,  catchint);
    signal(SIGHUP,  catchint);
    signal(SIGBUS,  catchint);
    signal(SIGTERM,  catchint);



    if(mknod("ppip1",010666,0)<0){
       cout<<"\nmknod pip1 \n";
    };
    if(mknod("ppip2",010666,0)<0){
      cout<<"\nmknod pip2 \n";
    };
    if ((tmp = open("ppip2",O_RDONLY|O_NDELAY))<0){
      cout<<"\n----------\n";
      exit(1);
    };

    if ((pdw = open("ppip2",O_WRONLY|O_NDELAY ))<0){
      cout<<"\n----------\n";
      exit(1);
    };

    if ((pdr = open("ppip1",O_RDONLY))<0){
      cout<<"\n----------2\n";
      exit(1);
    };

    while(1)
    {
       if((read(pdr,clnpid,5))>0)
          if((pd=fork())==0)      //child
          {     cout<<"new server  "<<getpid()<<"  created"<< endl;
                set_up_newpip(clnpid, rdpip,wrpip);
                while(!quit)
                    quit = console(&D,rp);
                close_up_exit();
          }
          else  if(pd <0)
                cout<< "error forking a new child \n";
          else if(pd>0)
                cout<<"parent....\n";
    }

}
```

pserver.C

```
//                    pserver.C

#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>
#include "L.C"
#include "itoa.h"
#define msgsize 80
struct rec {
        char name[30];
        char lname[30];
        char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      } ;

int pdr;
int pdw;
int rp;
int wp;
int pid[2];
char inbuf[msgsize];
char *datafile = "ldatafile";
char *ptrfile = "pharptrfile";
char *server;

char clnpid[80],rdpip[80],wrpip[80];
int tmp;
void  close_up_exit();

//*********************************************
int catchint(int signum)
{    int clnpid;
     cout<<"\n*****************************\n";
     cout<<"signal -----"<<signum;
     cout<<"\n*****************************\n";
     close_up_exit();
     exit(0);
};

//*********************************************
template <class keyType, class elemType>

class Dict: public List<keyType, elemType>{
  private:
     char *fname;
  public:
     Dict(){};
```

```
        Dict(char *);
        int  add(keyType k, elemType);
        elemType find(keyType k);
        void delet(keyType);
        void prnt();
        void lookup(){};
        void find(){};

        void operator[](){};
        ~Dict(){};
 };


 //--------------------------------
 //       Dict(filename)
 //--------------------------------
 template <class keyType, class elemType>
 Dict<keyType, elemType>::Dict(char *ptrfile)
 {
   keyType key ;
   elemType elem ;

   ifstream inFile (ptrfile,ios::in);
   cout<<"ptrfile="<<ptrfile<<endl;
   while(!inFile.eof()){
      inFile >> key;
      inFile >> elem;
      add(key,elem);
   }
   inFile.close();

};


//--------------------------------
//        add(key,elem)
//--------------------------------
template <class keyType, class elemType>
int Dict<keyType, elemType>::add(keyType k, keyType e){

          tree = insert(k,e,tree);
          return 1;
};
//--------------------------------
//display picture
//--------------------------------

void disply(char elem[])
{

   int n=0;
   char display[80];
   char *k;
   struct rec *r;
```

```
    long offset,position;
    r=new rec;
    FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

    offset = atoi(elem);
    if(offset == 0)
       cout<<"\n\n---------\a\a record not found\n\n";
    else{
       dfp=fopen(datafile,"rb");
       fseek(dfp,offset,0);
       fread((char *)  r, sizeof(struct rec),1,dfp);
       fclose(dfp);
       strcpy(display,"xv ");
       strcat(display,r->name);
       strcat(display," &");
       cout<<"\n++++++\n"<<display<<"\n+++++++++\n";
       system(display);
    }
};

//---------------------------------
//       find(key)
//---------------------------------
template <class keyType, class elemType>
elemType Dict<keyType, elemType>::find(keyType k){

      char *element = List<keyType, elemType>::find(k,tree);
      cout<<"\n before if.....\n";
      if (element != 0){
         cout<<"the element of key "<<k<<" is "<<element<<endl;
         //==============================
         //display(element);
         //==============================
         return element;
      }
      else{
         cout<<"key "<< k <<" does not exist \n";
         return ("notfound");
      };
};
//---------------------------------
//       delet(key)
//---------------------------------
template <class keyType, class elemType>
void Dict<keyType, elemType>::delet(keyType key){

      cout<<"removing "<<key<<endl;
          remove(key,tree);
};
//--------------------------------
//       print()
//--------------------------------
template <class keyType, class elemType>
void Dict<keyType, elemType>::prnt(){
```

```
        cout<<"-----------------------------\n";
            cout<<"\nprinting the sorted List\n\n";
        print(tree);
        cout<<"-----------------------------\n";
        printtree(tree,45);
        cout<<"-----------------------------\n";
};
//*************************************************
//-------------------------------------------------
template <class keyType, class elemType>
int console(Dict<keyType, elemType>  *D,int pd){

      int select;
      char choice[5];
      char key[20];
      char elem[20];
      char *element ;
      int quit=0;
      if((read(rp,choice,5))>0)
      {  cout<<choice<<"_________--------_________\n";
        select = atoi(choice);
        sleep(1);
        switch(select)
            {
              case 1 :  read(rp,key,20);
                 read(rp,elem,20);
                 D->add(key,elem);              //add
                            inbuf[0]='\0';
                            sprintf(inbuf,"%s  %s\n",key,elem);
                 write(pid[1],inbuf,msgsize);
                 break;

              case 2 :  read(rp,key,20);
                 element=(char *) D->find(key);   //find
                 write(wp,element,20);
                   break;

              case 4 :  cout<<"parent is printing\n";     //print
                    D->prnt();
                            break;

                 case  5 :  read(rp,key,20);
                 D->delet(key);                 //delete
                            inbuf[0]='\0';
                   sprintf(inbuf,"%s",key);
                 write(pid[1],inbuf,msgsize);
                 break;


              case 6 :  quit = 1;

            default:  break;
          }
```

```
        }
        return quit;
};
//-------------------------------------
void set_up_newpip( char *newpip , char  *rdpip, char *wrpip)
{
        int tmp;
        int quit=0;
        char *pid;

        cout<<"\ndebug2----the new pipe is: "<<newpip<<"\n";
        strcpy(rdpip,newpip);
        strcpy(wrpip,newpip);
        strcat(wrpip,"a");
        strcat(rdpip,"b");
        if ( (wp = open(wrpip,O_WRONLY )) <0){
          cout<<"\n----------\n";
          exit(1);
        };

        if ( (rp = open(rdpip,O_RDONLY) )<0 ){
          cout<<"\n\n----------1 \n";
          exit(1);
        };
        pid= itoa(getpid());
        cout<<"pid  "<<pid<<endl;
        cout<<"wrpip is: "<<wrpip<<endl;
        cout<<"rdpip is : "<<rdpip<<endl;
}

//-------------------------------------
void  close_up_exit()
{
        close(tmp);
        close(pdw);
        close(rp);
        close(wp);
        unlink(rdpip);
        unlink(wrpip);
        cout<<"server  "<<getpid()<<"  is killed"<< endl;
        cout<<"\n==== finished====\n";
        kill(atoi(clnpid),SIGTERM);
        exit(0);
}
//*************************************************
main(){

        int quit=0;

        int INT_signal();
        Dict<String,String>  D(ptrfile);

            char command[5];
            char rec[60];
```

```
        char name[60];
        char ptr[60];
    int  cmd;
        int pd;
    signal(SIGINT,  catchint);
    signal(SIGHUP,  catchint);
    signal(SIGBUS,  catchint);
    signal(SIGTERM,  catchint);



    if(mknod("ppip1",010666,0)<0){
       cout<<"\nmknod pip1 \n";
    };
    if(mknod("ppip2",010666,0)<0){
      cout<<"\nmknod pip2 \n";
    };
    if ((tmp = open("ppip2",O_RDONLY|O_NDELAY))<0){
      cout<<"\n----------\n";
      exit(1);
    };

    if ((pdw = open("ppip2",O_WRONLY|O_NDELAY ))<0){
      cout<<"\n----------\n";
      exit(1);
    };

    if ((pdr = open("ppip1",O_RDONLY))<0){
      cout<<"\n----------2\n";
      exit(1);
    };

    while(1)
    {
        if((read(pdr,clnpid,5))>0)
           if((pd=fork())==0)      //child
           {     cout<<"new server  "<<getpid()<<"  created"<< endl;
                 set_up_newpip(clnpid, rdpip,wrpip);
                 while(!quit)
                     quit = console(&D,rp);
                 close_up_exit();
           }
           else  if(pd <0)
                 cout<< "error forking a new child \n";
           else if(pd>0)
                 cout<<"parent....\n";
    }

}
```

lserver.C

```
//                    lserver.C

#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>
#include "L.C"
#include "itoa.h"
#define msgsize 80
struct rec {
        char name[30];
        char lname[30];
        char ssn[12];
        char addr[40];
        char docname[30];
        char test_type[20];
        char res[80];
      } ;

int pdr;
int pdw;
int rp;
int wp;
int pid[2];
char inbuf[msgsize];
char *datafile = "ldatafile";
char *ptrfile = "labptrfile";
char *server;

char clnpid[80],rdpip[80],wrpip[80];
int tmp;
void  close_up_exit();

//*********************************************
int catchint(int signum)
{    int clnpid;
     cout<<"\n*****************************\n";
     cout<<"signal -----"<<signum;
     cout<<"\n*****************************\n";
     close_up_exit();
     return(0);
};

//*********************************************
template <class keyType, class elemType>

class Dict: public List<keyType, elemType>{
   private:
     char *fname;
   public:
     Dict(){};
```

```
        Dict(char *);
        int  add(keyType k, elemType);
        elemType find(keyType k);
        void delet(keyType);
        void prnt();
        void lookup(){};
        void find(){};

        void operator[](){};
        void &operator[](){};
        ~Dict(){};
};


//--------------------------------
//       Dict(filename)
//--------------------------------
template <class keyType, class elemType>
Dict<keyType, elemType>::Dict(char *ptrfile)
{
  keyType key;
  elemType elem;

  ifstream inFile (ptrfile,ios::in);
  cout<<"ptrfile="<<ptrfile<<endl;
  while(!inFile.eof()){
     inFile >> key;
     inFile >> elem;
     add(key,elem);
  }
  inFile.close();

};


//--------------------------------
//       add(key,elem)
//--------------------------------
template <class keyType, class elemType>
int Dict<keyType, elemType>::add(keyType k, keyType e){

 tree = insert(k,e,tree);
 return 1;
};
//-----------------------------------
//display picture
//-----------------------------------

void disply(char elem[])
{

   int n=0;
   char display[80];
   char *k;
```

```
    struct rec *r;
    long offset,position;
    r=new rec;
    FILE *dfp, *pfp;  //dfp is data file ptr; pfp is ptr file ptr.

    offset = atoi(elem);
    if(offset == 0)
       cout<<"\n\n---------\a\a record not found\n\n";
    else{
       dfp=fopen(datafile,"rb");
       fseek(dfp,offset,0);
       fread((char *)  r, sizeof(struct rec),1,dfp);
       fclose(dfp);
       strcpy(display,"xv ");
       strcat(display,r->name);
       strcat(display," &");
       cout<<"\n++++++\n"<<display<<"\n++++++++\n";
       system(display);
    }
};

//---------------------------------
//       find(key)
//---------------------------------
template <class keyType, class elemType>
elemType Dict<keyType, elemType>::find(keyType k){

      char *element = List<keyType, elemType>::find(k,tree);
      cout<<"\n before if.....\n";
      if (element != 0){
         cout<<"the element of key "<<k<<" is "<<element<<endl;
         //==============================
         //disply(element);
         //==============================
         return element;
      }
      else{
         cout<<"key "<< k <<" does not exist \n";
         return ("notfound");
      };
};
//---------------------------------
//       delet(key)
//---------------------------------
template <class keyType, class elemType>
void Dict<keyType, elemType>::delet(keyType key){

      cout<<"removing "<<key<<endl;
  remove(key,tree);
};
//---------------------------------
//       print()
//---------------------------------
template <class keyType, class elemType>
```

```
void Dict<keyType, elemType>::prnt(){

        cout<<"-----------------------------\n";
   cout<<"\nprinting the sorted List\n\n";
        print(tree);
        cout<<"-----------------------------\n";
        printtree(tree,45);
        cout<<"-----------------------------\n";
};
//*************************************************
//-------------------------------------------------
template <class keyType, class elemType>
int console(Dict<keyType, elemType>  *D,int pd){

      int select;
      char choice[5];
      char key[20];
      char elem[20];
      char *element ;
      int quit=0;
      if((read(rp,choice,5))>0)
      {  cout<<choice<<"_________-------_________\n";
        select = atoi(choice);
        sleep(1);
        switch(select)
   {
      case 1 : read(rp,key,20);
                 read(rp,elem,20);
                 D->add(key,elem);              //add
                            inbuf[0]='\0';
                    sprintf(inbuf,"%s  %s\n",key,elem);
                 write(pid[1],inbuf,msgsize);
                 break;

      case 2 : read(rp,key,20);
                 element=(char *) D->find(key);   //find
                 write(wp,element,20);
           break;

      case 4 : cout<<"parent is printing\n";     //print
            D->prnt();
                  break;

         case  5 : read(rp,key,20);
                 D->delet(key);                 //delete
                            inbuf[0]='\0';
                    sprintf(inbuf,"%s",key);
                 write(pid[1],inbuf,msgsize);
                 break;


               case 6 : quit = 1;

           default: break;
```

```
            }
        }
        return quit;
};
//-------------------------------------
void set_up_newpip( char *newpip , char  *rdpip, char *wrpip)
{
        int tmp;
        int quit=0;
        char *pid;

        cout<<"\ndebug2----the new pipe is: "<<newpip<<"\n";
        strcpy(rdpip,newpip);
        strcpy(wrpip,newpip);
        strcat(wrpip,"a");
        strcat(rdpip,"b");
        if ( (wp = open(wrpip,O_WRONLY )) <0){
          cout<<"\n ---------\n";
          exit(1);
        };

        if ( (rp = open(rdpip,O_RDONLY) )<0 ){
          cout<<"\n\nError opening the  "<<rdpip<<" \n";
          exit(1);
        };
        pid= itoa(getpid());
        cout<<"pid  "<<pid<<endl;
        cout<<"wrpip is: "<<wrpip<<endl;
        cout<<"rdpip is : "<<rdpip<<endl;
}

//-------------------------------------
void  close_up_exit()
{
        close(tmp);
        close(pdw);
        close(rp);
        close(wp);
        unlink(rdpip);
        unlink(wrpip);
        cout<<"server  "<<getpid()<<"  is killed"<< endl;
        cout<<"\n==== finished====\n";
        kill(atoi(clnpid),SIGTERM);
        exit(0);
}
//*************************************************
main(){

        int quit=0;

        int INT_signal();
        Dict<String,String>  D(ptrfile);

 char command[5];
```

```
    char rec[60];
    char name[60];
    char ptr[60];
        int  cmd;
    int pd;
        signal(SIGINT,  catchint);
        signal(SIGHUP,  catchint);
        signal(SIGBUS,  catchint);
        signal(SIGTERM,  catchint);



        if(mknod("lpip1",010666,0)<0){
           cout<<"\n mknod pip1 \n";
        };
        if(mknod("lpip2",010666,0)<0){
          cout<<"\n mknod pip2 \n";
        };
        if ((tmp = open("lpip2",O_RDONLY|O_NDELAY))<0){
          cout<<"\n----------\n";
          exit(1);
        };

        if ((pdw = open("lpip2",O_WRONLY|O_NDELAY ))<0){
          cout<<"\n\n ---------  \n\n";
          exit(1);
        };

        if ((pdr = open("lpip1",O_RDONLY))<0){
          cout<<"\n\nError openning pip1 \n\n";
          exit(1);
        };


        while(1)
        {

           if((read(pdr,clnpid,5))>0)
              if((pd=fork())==0)       //child
              {      cout<<"new server  "<<getpid()<<"  created"<< endl;
                     set_up_newpip(clnpid, rdpip,wrpip);
                     while(!quit)
                         quit = console(&D,rp);
                     close_up_exit();
              }
              else  if(pd <0)
                     cout<< "error forking a new child \n";
              else if(pd>0)
                     cout<<"parent....\n";
        }

}
```

dstart.C

```
#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>


void  close_up_exit();

//*********************************************
int catchint(int signum)
{    //system("cls");
     exit(0);
}
//*************************************************
void main()
{
       int pid;
       int i;
       char st[100][100];
       char str[80],strn1[80],strn2[80];
       signal(SIGINT,  catchint);
       signal(SIGHUP,  catchint);
       signal(SIGBUS,  catchint);
       signal(SIGUSR1, catchint);
       signal(SIGTERM,  catchint);
       signal(SIGQUIT,  catchint);
       signal(SIGKILL,  catchint);
       system("cls");
       cout<<"\n connecting to main server \n\n";
       pid=fork();

       if (pid== -1)
         cout<< " something wrong\n";

       if (pid>0 ){ // parent
          while(1){
            system("ping axon>tpf");
            ifstream inFile ("tpf",ios::in);
            inFile >> str ;
            inFile >> strn1 ;
            inFile >> strn2 ;
            inFile.close();
            //cout<<str<<strn1<<strn2<<endl;
            if ((strcmp(str,"no"))==0) {
               kill(pid,SIGUSR1);
               execl("dstartb","startb",(char *)0);
             }

          }
        }
       if (pid == 0 ) // child
        {    //cout<<"at child pid= "<<getpid()<<"\n";
```

```
            system("rsh isdn  doct or cheetah:0.0");
        }
}
```

dstartb.C

```
#include <stdio.h>
#include <string.h>
#include<signal.h>
#include <String.h>
#include <fstream.h>
#include <fcntl.h>
void  close_up_exit();

//*********************************************
int catchint(int signum)
{
    exit(0);
}
//*************************************************
void main()
{
       int pid;
       int i;
       char st[100][100];
       char str[80],strn1[80],strn2[80];
       signal(SIGINT,  catchint);
       signal(SIGHUP,  catchint);
       signal(SIGBUS,  catchint);
       signal(SIGUSR1, catchint);
       signal(SIGTERM,  catchint);
       signal(SIGQUIT,  catchint);
       signal(SIGKILL,  catchint);
       system("cls");
       cout<<"\nconnecting to backup server \n";
       pid=fork();

       if (pid== -1)
         cout<< " something wrong\n";
       if (pid>0 ){ // parent
          while(1){
            system("ping isdn>tpf");
            ifstream inFile ("tpf",ios::in);
            inFile >> str ;
            inFile >> strn1 ;
            inFile >> strn2 ;
            inFile.close();
            //cout<<str<<strn1<<strn2<<endl;
            if ((strcmp(str,"no"))==0) {
               kill(pid,SIGUSR1);
               execl("dstart","dstart",(char *)0);
             }
          }
       }
       if (pid == 0 ) // child
       {    //cout<<"at child pid= "<<getpid()<<"\n";

            system("rsh isdn  doctor cheetah:0.0");
       }
}
```