"BULLETIN BOARD SERVICE IN NOVELL NETWARE"

PROJECT REPORT

P-233

SUBMITTED BY:

Balasundar Raj G.
Kadhirvelu P.
Rajesh G.

Under the Guidance of:
Mr. G. Balasubramanian B.E., M.S.

Submitted in partial fulfilment of the requirements for the award of the Degree of
Bachelor of Engineering
in ***Computer Science and Engineering*** *of the*
Bharathiar University*, Coimbatore.*

Department of Computer Science and Engineering
Kumaraguru College of Technology
Coimbatore - 641 006
1995-96.

Kumaraguru College of Technology

Coimbatore -641 006.

Department of Computer Science and Engineering

Certificate

This is to Certify that the Report entitled

"BULLETIN BOARD SERVICE IN NOVELL NETWARE"

has been submitted by

Mr/Ms Balasundar Raj G, Kadhirvelu P, Rajesh G

in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in the Computer Science and Engineering Branch of the Bharathiar University, Coimbatore - 641 046 during the academic year 1995-'96.

[illegible]
(Guide)

(Head of Department)

Certified that the Candidate was Examined by us in the Project Work Viva-Voce Examination held on ________________ *and the University Register Number is* ________________

(Internal Examiner)

(External Examiner)

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

We wish to thank our beloved principal **Dr.S.Subramanian** for his kind patronage in providing us the infrastructural facilities for carrying out this project.

We sincerely thank **Prof. P.Shanmugam M.Sc(Engg), M.S.(Hawaii),** for his kindness, support and expert advice, without which this project would never have been a possibility.

P-233

We would like to express our sincere thankfulness and profound gratitude to our project guide, **Mr. G. Balasubramanian B.E., M.S,** for his invaluable advice. We really appreciate his constant efforts on motivating and encouraging us and for his patience in editing this project.

Finally, we would like to thank all our colleagues at the Department of Computer Science and Engineering, Kumaraguru College of Technology, for their help throughout our project work.

SYNOPSIS

A Bulletin Board Service is an electronic club where individuals or institutions or organisations share information and data pertaining to varied topics by means of databases, personal information, special interest groups, discussion forums etc. Any member of this club can write and submit articles on any topic and in turn read articles submitted by other members. This project is an attempt, to provide a Bulletin Board Service in Novell Netware environment. The Software has been implemented in Turbo C using IPX/SPX primitives in a local area network under DOS platform.

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION TO LOCAL AREA NETWORK

The term "Computer Network" refers to an interconnected collection of autonomous computers. Two computers are said to be interconnected if they are able to exchange information. The connection need not be via a cable; lasers, microwaves, and communication satellites can also be used. The important features of computer networks are

(a) To provide shared access to information

(b) Permit movement of data

(c) High reliability

(d) Save money

(e) Serves as a communication medium for widely separated people.

The local area network is a form of computer network. Local area network are data communication system that span within a small area .

1.1.1 Network Topologies

Based on the working environment the network topology differs. A local area network usually has a very regular form that is either a star or a bus or a ring. Each topology is best suited to particular media types, routing strategy and reliability characteristics.

The Star

The star is a convenient topology because it permits easy routing. The central node knows the path to other nodes. A star configuration provides a central control point for easy access to the network giving priority status to selected nodes and for performing network maintenance and testing. The drawback of the centralisation of control is that the centre node should be exceptionally reliable and have the computational capacity to route all the network traffic.

The Ring

The ring topology is used for transmission media that are simpler or difficult to tap, as all transmission occur in one direction. The ring requires no routing, as each node passes on a message. Ring system have the unique feature, that they can be arranged to provide the reception verification. Every node of a ring must work in order for the network to function correctly. This drawback can be avoided by designing practical rings which may be of Ring network with bypass relays or Ring network with bypass relays between nodes or ring network with wire centres.

The Bus

A Bus requires a full duplex medium i.e., one in which signals can flow in either direction. Nodes associated with a bus do not perform routing message forwarding. The bus is a broadcast medium in which all nodes receive all transmission. A major advantage of the bus is that nodes can fail and traffic can still be passed over the network. Also the addition of nodes can often be accomplished with disrupting network traffic. The drawback is that the impedance irregularities caused by the installation of taps will cause signal reflections that can interfere with data transmission.

1.1.2. COMPONENTS OF A LAN

Basic LAN hardware components consist of Network adapter cards, cables and either access units or repeaters. Depending on its type and complexity, the network may also include bridges, routers, Concentrators, baluns, hubs and transceivers.

Network adapter cards

Network adapter cards can be either collision sensing or token passing. Both kinds of adapters have sufficient on board logic to know when it is okay to send a frame and to recognise when the frames they receive are intended for them. Steps during sending or receiving a frame are:

1. Data Transfer
2. Buffering
3. Frame formation
4. Cable access
5. Parallel and serial conversion
6. Encoding and decoding
7. Sending and receiving impulses.

1.1.3. Cabling System

* BULLETIN BOARD SERVICE

Generally cabling system for LANs vary widely in their outward appearance, characteristics, intended purpose and cost. The three most popular ways of tying Computers together on a LAN are

* The IBM Cabling System
* The AT & T premises Distribution System
* The DEC Cabling concept called DEC connect

The Cabling system can be categorised into three distinct cable types.

* Twisted pair (Shielded and unshielded)
* Coaxial cable (thin and thick)
* Fibre - optic cable

Twisted pair:

Twisted pair are the insulated wires with a minimum number of twists per foot. Twisting reduce electrical interference (attenuation). Shielded refers to the amount of insulation around the wire and therefore its immunity to data - corruption errors.

Coaxial Cable:

The Coaxial Cable consists of a copper conductor surrounded by insulation, which in turn is encircled by a tube shaped conductor of solid copper, aluminium. The outermost tube and the centre Conductor share the same axis of curvature, hence the term "Co axial". Coaxial cables are used in baseband system as well as broadband system.

* BULLETIN BOARD SERVICE

Fibre-optic cable:

Fibre - optic cable uses light rather than electricity to carry information. Although it sends data over huge distances at high speeds Fibre-optic cable is expensive and difficult to install and maintain.

It consists of a hollow core fibre whose diameter is measured in microns, surrounded by a solid cladding which in turn is covered by protected sheath. The light source for fibre - optic cable is a light emitting diode (LED); information usually is encoded by varying the intensity of the light. A detector at the other end of the cable connects the received signal back into electrical impulse. There are two types of fibre, single mode and multimode. The more expensive single mode has a smaller diameter and can carry signals for a greater distance.

File Server:

A file server is a place to store the files that we want to share among the PCs. We can turn one of the PCs into a file server or use another computer as the Server either way a server provides

* Fast access to the files

* Capacity to hold files and records for many users

* Security for the files.

* Reliability.

We can often set up a file server so that it can be used as a workstation at the same time as a file server. Such servers are termed as non-dedicated file servers. On the other hand a file server serving the purpose of file storage (management) is termed as a dedicated server.

1.2. NETWORK PROGRAMMING

Pc - Pc Communication concepts:

The OSI Model defines LAN communication functions as seven layers. The seven layers are.

1. Physical (lowest) layers:

This part of the OSI Model specifies the physical and electrical characteristics of the connections that make up the network.

2. Data Link:

At this stage of processing the electrical impulses enter or leave the network cable. Bit patterns, encoding methods, and tokens are examples of elements known to this layer. At this stage, errors are detected and corrected. Because of its complexity, the datalink layer is divided into a Medium Access Control layer and a Logical Link Control layer. The MAC layer manages network access and network control. The LLC layer, operating at a higher level than the MAC layer, sends and receives the user data message themselves.

3. Network Layer:

This processing step switches and routes the packets as necessary to get them to their destinations. It is the layer responsible for addressing and delivering message packets.

4. Transport Layer:

When more than one packet is in process at any time, this layer controls the sequencing of the message components and also regulates traffic flow. If a duplicate packet arrives, this layer recognises it as a duplicate and discards it.

5. Session Layer:

The functions in this layer let applications running at two workstations co-ordinate their communications into a single session. A session is an exchange of message between two workstations. This layer supports the creation of the session, the management of the packets sent back and forth during the session, and the termination of the session.

6. Presentation Layer:

When IBM, Apple, DEC and Burrough computers all want to talk to each another, some translating and byte recording is needed. This layer converts data into a machine's native internal numeric format

7. Application (highest) layer:

This is the layer seen by application program, and therefore is also the interface to the OSI Model that the programmer sees. A message to be sent across the network enters the OSI Model at this point, travels downward the first layer (the physical layer) Zips across to the other work station and then travels back up the layers until the message reaches the application on the other computer through its Application layers.

1.3 HIGHER-LEVEL PROTOCOLS

The protocols discussed next are IPX, SPX and NETBIOS. Each of these is directly programmable by your software. The actual data message that fly around the network come from either your software or from the shell /redirector software that shuttles DOS file-service requests to the file server and back. Here IPX is discussed in detail.

1.3.1. Datagrams and Sessions:

The two types of PC to PC or PC to server communication are datagrams and sessions. A datagram is a message never acknowledged by the receiver, if message delivery must be verified, the receiver must supply a return message.

A session is a logical connection between two workstations in which message reception is guaranteed. Datagrams can be sent at will. For message to be sent during a session, however, more work must be done: the session must be established, data messages sent and received, and at the end of the dialog the session must be closed.

1.3.2. IPX - a Datagram Service:

The underlying protocol in Novell Netware, Internetwork Packet Exchange(IPX) is a close adaptation of a protocol developed by Xerox corporation, the Xerox Network Standard (XNS). IPX supports only datagram message.

* BULLETIN BOARD SERVICE

Corresponding to the Network Layer of the OSI Model, this protocol performs addressing, routing, and switching to deliver a message to its destination. IPX is speedier than the session-oriented SPX protocol. Datagram delivery is not guaranteed, but Novell indicates that IPX packets are correctly received about 95 percent of the time.

Using IPX is safe and reliable because every such request from a workstation requires a response from the file server. Until the proper response is received, the shell /redirector never assumes that a fileserver has processed a file service packet.

If we use Novell Netware, we already have IPX. Depending on the version of Netware we have, we may already have SPX as well.

1.3.3. IPX Destination Address:

The Destination address in an IPX packet is an internetwork address, which consists of three components:

(i) Network number

(ii) Node address.

(iii) Socket

* Network numbers identify each segment of a multiserver network. System Administrators assign the numbers when the Novell Netware fileservers are initialised.

* Node address uniquely identifies each network adapter card.

* Socket represents the destination application itself, running on the target workstation. For IPX packets, the destination address can

* BULLETIN BOARD SERVICE

contain a group (broadcast) address; if our application needs to communicate with a particular workstation, we can use services within IPX to find that work station's destination address. We send IPX packets by specifying a destination address (network, node and socket), but we receive them by specifying a socket.

1.3.4. IPX Program Services

We can manage Pc to Pc communications by using a set of service that IPX offers to your software. These services rely on the following three data structures that we construct and pass to Ipx:

* The IPX Header, consisting of the first 30 bytes of the IPX packet discussed later.

* The data record of message that you want to sent or receive

* An Event Control Block (ECB) that is not actually transmitted but contains information about a particulars IPX operation that we initiate.

1.4. INITIALISATION AND TERMINATION FUNCTIONS

When two PCs on a network want to send message records back and forth using IPX, the application on both workstations first opens a socket using the IPXOpenSocket function. The socket number that is open on workstation A must be known to the application running on workstation B and viceversa. Both workstations needs to know the destination address of the other workstation. Socket numbers are easily determined: you simply establish conventions for the sockets you use. However your

* BULLETIN BOARD SERVICE

applications does not automatically know the network number and node address - what you typically know is the other workstation's user ID.

To translate user ID into network number and node address, use the GetObjectConnectionNumbers and the GetInternetworkAddress IPX functions at the beginning of your program. However, the Netware lets a single user ID log on at several workstations simultaneously. The usual way to handle this situation is by using the first item from the list that is returned to you and ignoring any subsequent items in the list.

When finished sending and receiving message records, the two workstations close the opened sockets by calling IPXCloseSocket.

1.4.1. Sending and receiving packets

After you have opened a socket at workstation A and determined the network number, node address, and socket number of workstation B, you are ready to send and receive message records. IPXSendPacket and IPXListenForPacket are used to do the sending and receiving. Each time you call IPXSendPacket, however you must supply an additional data item, the immediate address field. If the message packet must cross a bridge to reach its destination, the immediate address field is the node address of the bridge. You call the function IPXGetLocalTarget to determine the value to be placed in the immediate address field. If the packet does not need to cross a bridge, IPXGetLocalTarget will return the node address of the destination workstation.

IPX does not wait for the packet to be sent or received before returning to your application program; the protocol only initiates the operations. The actual sending and receiving occur in the background. You can handle these operations in one of two ways: either the application can go into a loop that repeatedly checks to see whether the send or receive operations is completed, or the application can supply IPX with the address of a routine that IPX will execute when the operation finishes.

1.5 MISCELLANEOUS FUNCTIONS

Outstanding IPX operations (events) can be cancelled with the IPXCancelEvent call. Your application can call IPXScheduleEvent to ask IPX to schedule a send or receive event to occur at a later moment. Especially while you can call IPXRelinquishControl to give IPX control of the CPU; this gives IPX a chance to "breathe" while you wait for the operation to complete.

1.6 IPX PACKET FORMAT

The IPX packet format is shown in the figure below:-

	LENGTH IN BYTES
CHECK SUM	2
LENGTH	2
TRANSPORT CONTROL	1
PACKET TYPE	1
DESTINATION NETWORK	4
DESTINATION NODE	6
DESTINATION SOCKET	2
SOURCE NETWORK	4
SOURCE NODE	6
SOURCE SOCKET	2
DATA PORTION	0-546

Check sum

This two byte field is a holdover, because the lowerlevel protocol always performs errorchecking, you never need to set this field.

Length

The size of the complete packet including both the IPX packet header and the data portion is expressed by this 2-byte field. The smallest packet length is 30 bytes and the largest is 576 bytes.

Transport Control

An IPX packet can cross as many as 16 netware bridges. IPX sets this 1-byte field to O when the packet is originally transmitted and then increments the field each time a bridge passes along the packet. If the count reaches 16, the packet is discarded, in normal situation, of course, this is not a concern. IPX sets and uses this field.

Packet Type

Various types of packets are defined for various purposes. an application that sends IPX packets should set this 1-byte field to a value of 4.

Destination Network

This is a 4-byte field that, as mentioned previously identifies the network on which the intended receiver workstation is located.

Destination Node

These six bytes identify the target workstation by its unique physical address.

Destination Socket

This field contains a 2 - byte socket number. The socket must have been opened by the application running on the destination workstation. You can ask Novell to assign a particular socket number to your application or if you want, you can use dynamic socket numbers, which are in the range 0X 4000 to 0X 8000 Source Network.

The Network number of the originating workstation, this 4 - byte field is set by IPX.

* BULLETIN BOARD SERVICE

Source Node

The physical address of the network adapter card in the originating workstation, this 6 - byte field is set by IPX.

Source Socket

The packet is sent through an open socket, which IPX sets in this field.

Data portion

This field the data record or message your application wants to send - can be from 0 to 546 bytes long.

CHAPTER II

BULLETIN BOARD SERVICE (BBS)

2.1. INTRODUCTION

A Bulletin Board service is an Electronic club where individuals institutions or organisations share information and data pertaining to varied topics by means of database, personal information, special interest groups, discussion forums etc. BBSes also contain huge libraries of shareware
software, which are available to download.

Generally these Bulletin Board systems are run by Computer hobbyists, so most of the time they are free or very economical. It is because of this virtue of the BBS'es to be free, they are almost always the first and foremost choice of any computer modem or computer novice to call and seek help. Also, surprisingly in this world of business and opportunities, most of the users on the BBS world are very helpful lot. This is also one of the major reasons for the widespread popularity of the Bulletin Board systems world-wide. In America alone there are over 55,000 BBSes running.

Practically speaking accessing BBS is like subscribing to a collection of electronic magazines. These "magazines", called newsgroups or forums, are devoted to particular topics ranging from politics, philosophies, science and recreation activities. Any

BBS user can write and submit articles on any topic and, in turn read articles submitted by other users.

2.2. BBS INTERNALS

2.2.1. WHAT MAKES A BBS?

A bulletin board system is basically a computer connected with a modem and a line on it. This software automatically picks up the phone on receiving a ring on the line, and then answers the call by a login to the caller. The software then presents the users with a wide range of menus that the users can select using this keyboard. Many BBSes also offer the mouse support, meaning that the callers can use their mouse to select a menu option.

Since these generally operate from some computer hobbyist's bedroom or study room, the BBSes generally reflect the personal tilt of sysop.

A sysop is an acronym or short form of system operator. In other words a sysop is the guy who runs the BBS. It is generally his machine, unless borrowed of course, that is dedicated to the BBS, it is generally his time, that go into the BBS, and it is his craze to run the board that keeps it running.

2.2.2. TYPES OF BBSes

The popularity chart of bulletin board services is increasing by the hour, with this increase the complexity of the BBS, the facility offered and the demands of the users have also increased manifolds.

However we can still divide the BBSes into the following three categories.

1. General
2. Company sponsored
3. Specialist

In our country most of the BBSes fall under the first, that is the general category. Since most of the BBSes (barring a few like Livewire, ECTC Net, CIX, etc..) are quite new, right now they are concentrating on stabilising and streamlining the operations on the BBS system, once they do settle down they will then diversify on the contents that they offer. Already many BBSes are showing signs of specialisation. One BBS offers extensive on-line material on astronomy, the other on lyrics and music etc. The second category is the company sponsored, now these are the ones that are more or less non - existent in our country. There has been the growth of one of the major publication unit going in for BBS. There are many other organisations whose BBSes are in the pipeline, including many financial services, many telecom companies and many hardware manufacturers selling datacom related products.

2.2.3. HOW TO CALL A BBS

Well, you have a modem, a computer with communication software loaded on it and a telephone line, that's all you need to call a BBS. Calling a BBS is no specific task other than matching the BBS parameters. Most of the BBSes will advertise their speed, their connect data structure and the types of compression, error correction that it support.

2.2.4. THE NEWS

Most of the BBSes have a news file, which gets displayed when you log on to the BBS, this is meant to inform the user of the latest development on the board. The news section sometimes is also used to convey every feelings on a particular topic, usually by sysop. Ex. the shoemaker levy comet strike news were flashed on the BBS news section at almost all BBSes. This news file also tells you of the latest bulletins on the board, a recent achievement of the board or a recent request for some information. The main purpose of this screen is to update the users of the recent happenings, in most cases you are requested to merely press <ENTER> after you have read the contents !

2.2.5. THE MAIN MENU

Each of the BBSes have a different structure of their main menu but basically you are given a list of options from which you are required to select an option of your choice.

Pressing the relevant alphabets on your keyboard you will be guided through the bulletin board. Assuming you have to send a mail to someone on the BBS, quite simple actually. The options can be selected in two ways. In the first case when the "Hotkey is Active" you will have to simply :

Press the relevant alphabet key to take you to the relevant area. However it is also quite possible that you will have to ?? :

Press the relevant alphabet key

followed by <Enter> to get into the relevant menu

2.2.6. EXCHANGING MAILS AND MESSAGES

One very important feature of almost all the BBSes is the facility to send and receive mails and to participate in a conference areas. Using a BBS you can send mails to your dear ones. Now there will be two kind of mails that you want to send :

1. Private

This mail will be specially for the addressed person only. Nobody else will be able to read mail of this sort. You might like to wish your friend on her 18th birthday or send a stinker to your competitor who is also a member of the same BBS.

2. Public messages

This mail is also sent to the addressed person, however this mail can be read by anybody on the BBS. You would probably like to tell everybody that your company has launched a new product while you are addressing a specific query from somebody. These types of mails are also used to create conference areas. In the conference areas generally there is a discussion on how to make the best combination of Rosgullas and Redwine on one of the BBSes, now obviously the more people there are to discuss on a particular topic, more will be the fun.

TYPES OF AREAS

When we discuss the areas we also discuss the types of forums.

1. Local Forums

Any message that you see in this area is read by the people of the BBS only. Here you will post these messages that are specific to this BBS only. Supposing you are not able to access some areas of BBS and you want to leave a mail for the sysop, then you will use this forum to discuss the issue with the sysop or other users on this BBS.

2.Echo Areas

Most BBSes now are connected to each other through some net. This simply means that sysops of these BBSes have come to an understanding that they will share the messages with each other. The message forums that are shared by the BBSEs are known as echo areas. Any message that you leave here or see here is available at all the BBSes that are part of the net.

2.3. MORE ABOUT BBS

2.3.1. WHAT CAN I DOWNLOAD?

Anything that relates to computers in terms of information and software can be downloaded from the BBS. You will find most of the BBSes around the country well stocked with lot of files.

1. The first thing and the most numerous in number is the shareware that you can download from the BBSes, these are

available by the dozens and you will find one for every application that you can possibly think of. You have shareware files for word-processing, desktop, graphics viewer, animator, morphings, autocad, educational utils, even stuff telling you how to make the best rosgullas with wine!!.

2. Graphics Files Another very popular thing you can pick up off any BBS is the huge collection of the graphics file, generally these are in the format of GIF or JPGS, to view them you need a special viewers which will also be available from the BBS from where you downloaded the GIFS.

3. Information Details

There is a huge list of information that you can pick up off the BBSes. You will find a textfile explaining to why is it better to install OS/2 2.1 and why not Win 3.1., about how to set-up your very own BBS, how to make a soundblaster card yourself and so on. You can even download files that describe the internet mailing procedure etc.

2.3.2. HOW TO DOWNLOAD

Most of the BBSes have huge collections of files ranging from all topics on computers to utilities to graphics etc. In order to make the task of selecting a file simple, the files are placed in different areas under specific headings, you may find all games files in a separate area, all graphics in a separate area, all programmers tools in a different area. So before you wish to download any file, you have to first go into the relevant area and do the download.

BBSes also give you the option of listing the files that are available for download, each of the file in the file listed is also commented so that the users know of what file they are about to download. For the sake of downloading a file, most of the BBSes give a specific option to the users so that the users can download the files. Once the download option is selected then the next question asked by the BBS is the protocol that is going to be used for the transfer of files. Depending on the line and modem connection you can select a specific protocol, if you are not sure about which option to select, use zmodem, it is quite a popular protocol by virtue of its crash recover facility. The name of the file you want to download comes next, enter the complete name of the file that you want to download. Follow this with the PgDn key at your end, which will initiate the download option of your communication program side. Many communication software gives you the statistics of the download in progress, the time it will take for the total file, time elapsed in this download, time left etc.,

2.4 PROBLEM DEFINITION

INTERNET the world's biggest network connects large number of countries. India is just beginning to get connected to this vast cyberworld. Bulletin Board service, one of the important facility in INTERNET has many features like involving the users in the topics of their interest, it provide exposure to the hottest technology in the world to all it users quickly and simultaneously. Here we have made an attempt to provide a basic BBS which can be enhanced into a full fledged one.

CHAPTER III

SYSTEM DESIGN

3.1 OVERVIEW

Our BBS is an electronic club with three forms. The forums are

1. Bulletin
2. Technical
3. Fun

In bulletin the college news, announcements and lots of other information about the college is available.

Technical forum is for the techie guys to know latest state of art technology in various fields of science.

Fun is the most cool part of our BBS which entertains its members with a lot of comedy.

The user of this BBS has basically two options, one, to add his article into the forum of his choice and two, to select an article from a forums of his choice and read. The user can use an editor to type in his article.

3.2. Modules

The BBS software consists of two parts.

1. Server
2. Client

Server

The server is the node in the network where the database of the different forums are placed in order that they are centralised and can be accessed by any user. A user who wants to use the BBS. request this server to initiate and complete the desired job. The server first receives the header packet from the client. This header packet gives the server the operation required by the client. The information in the header packet is interpreted and acknowledgement is sent to the client to be ready to receive the finished operation. Then transfer of file from server to client or vice versa is performed according as to view the article or add an article to a forum respectively. An acknowledgement is sent from the receiver side everytime it receives the correct data packet. The sending side sends the next packet only after receiving this acknowledgement packet. Once the operation is complete the server routine is again ready to receive the next header packet, to perform the next operation.

Client

The client part is the one that interacts with the actual user. The user selects his option, whether to add an article or view an article accordingly, the request is sent to the server through the header packet specifying the operation and information of the article according to the option selected. The transfer of file in the network is using IPX, hence the file transfer has to the asynchronous i.e. for every data packet received in the destination side, an acknowledgement packet is sent to the source side.

3.3. Data Structures

Packet format that is used to send is shown below.

IPXHeader	Hdr
Char	srcid
Char	Fnam
int	state
int	pc
int	fsz
int	oper
int	flag
char	data

The above structure is the packet that is used to transfer a file from the server to client or vice versa. The different fields are explained below.

IPXHeader - 30 byte long, that contains socket number, destination address, packet type & packet size.

srcid - a character array that contains the id of the sender.

Fnam - a character array that stores filename i.e. artic.e name.

state - specifies whether the packet is a header packet, data packet or acknowledgement packet.

pc - specify to the destination node the number of packets that are to be transferred from the source.

fsz - Specifies the file size

oper - Used to specify the server the type of operation requested by the client.

flag - to indicates whether the server is busy or not.

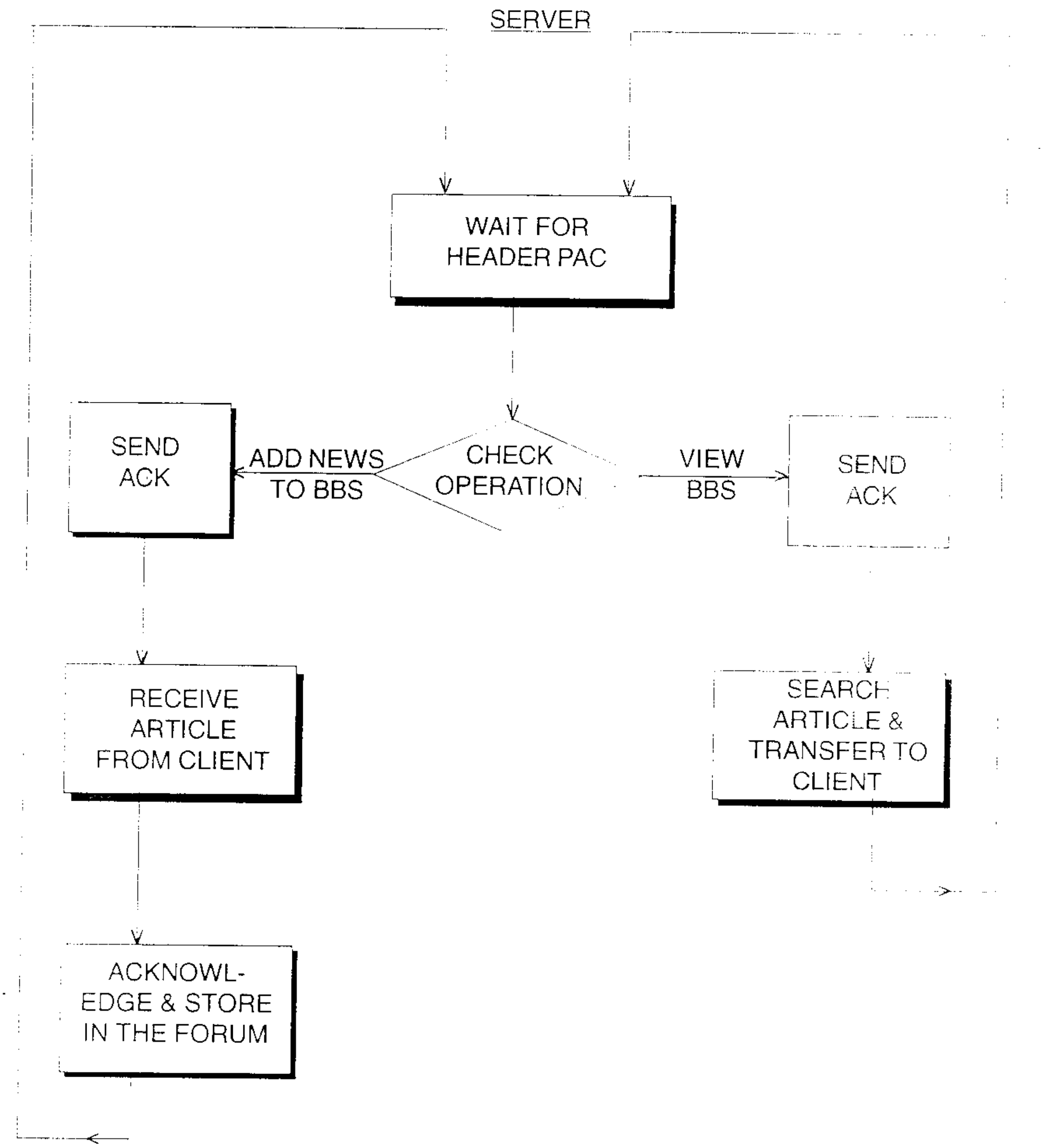
Data - character array which takes file from the source to the destination node.

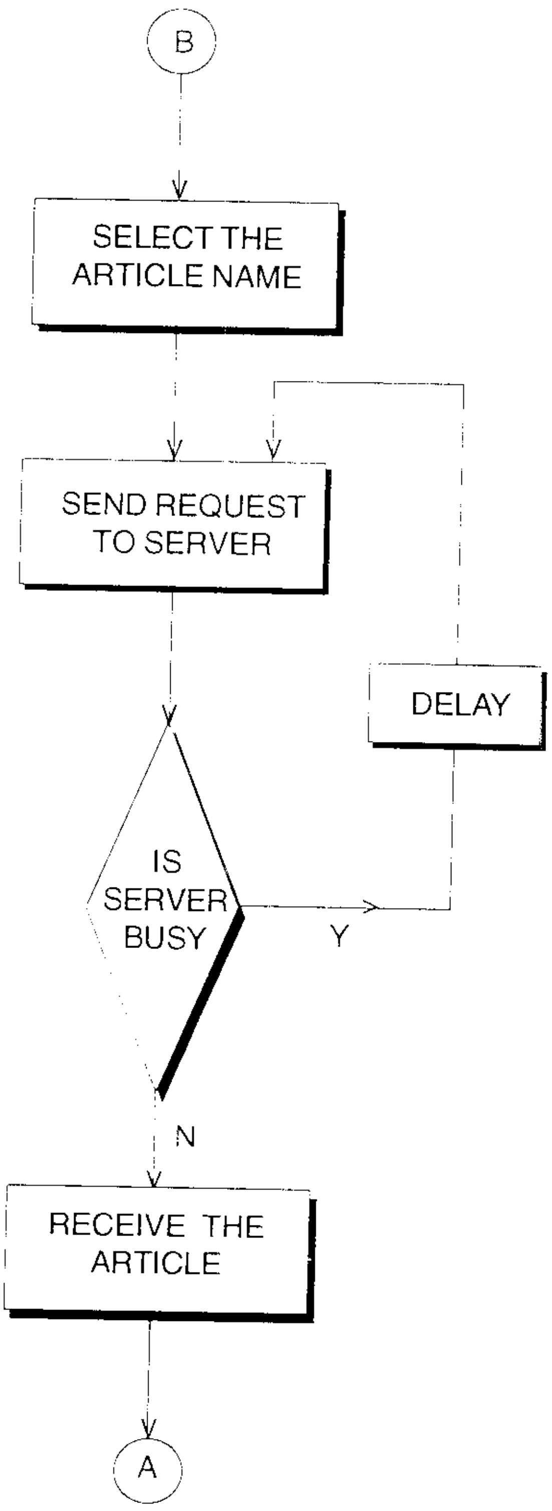
Article header that is used to store for each article is shown below.

char	id
char	tit
char	des
int	day
int	mon
int	yr
int	hr
int	min
int	sec

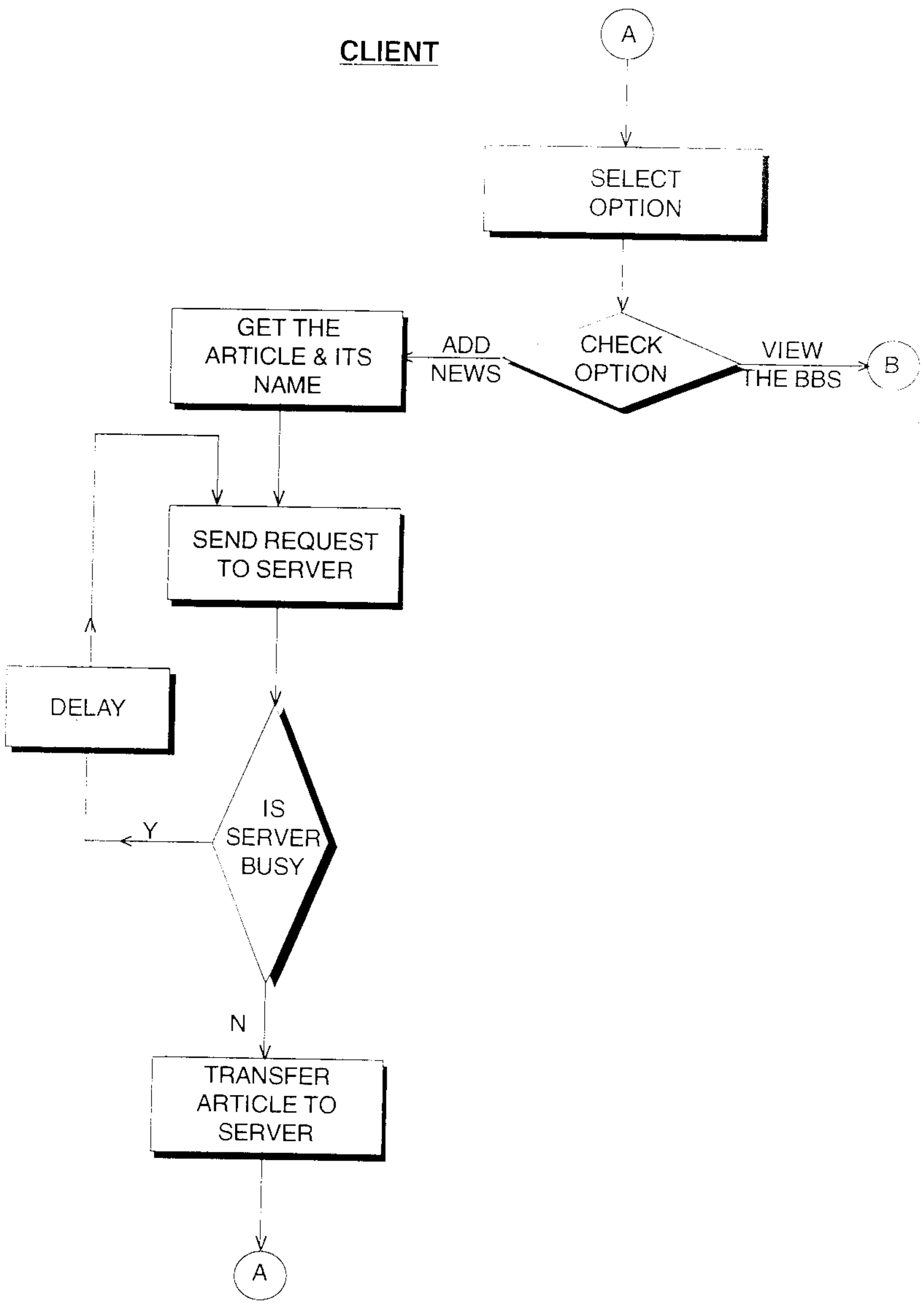
The above structure is used to maintain the directory of articles in different forums. The id specifies the senders name, tit - the articles title, des description of the article. Then the data & time when the article was written is stored. The list of these structure is retrieved for the particular forum requested whenever the client chooses the option to view an article. From this list the user chooses the article and sends the request to the server.

SYSTEM FLOW:-


SERVER
WAIT FOR HEADER PAC
CHECK OPERATION
ADD NEWS TO BBS
VIEW BBS
SEND ACK
SEND ACK
RECEIVE ARTICLE FROM CLIENT
SEARCH ARTICLE & TRANSFER TO CLIENT
ACKNOWL-EDGE & STORE IN THE FORUM


B
SELECT THE ARTICLE NAME
SEND REQUEST TO SERVER
IS SERVER BUSY
Y
DELAY
N
RECEIVE THE ARTICLE
A

CLIENT


A
SELECT OPTION
CHECK OPTION
ADD NEWS
VIEW THE BBS
B
GET THE ARTICLE & ITS NAME
SEND REQUEST TO SERVER
IS SERVER BUSY
Y
DELAY
N
TRANSFER ARTICLE TO SERVER
A

CHAPTER IV

IMPLEMENTATION DETAILS

BBS software is developed using TURBO C under DOS platform. The server routine is run in a node and it is always kept ready to receive a header packet from the client. The node that is running this server routine is always ready to receive the header pac. This may put the server in a monotonous job when there are no clients accessing the server.

As a client a user selects his option whether to add a new article to a forum or to read an article from a forum. If he chooses to add a new article, the user is taken inside the NORTON editor to type & edit his article after getting article's name and description.

The article header is then formed with source id and the request is sent to the server. Once the server is notified about the client's request then after receiving the acknowledgement from the server, the client starts sending the article in data packets of 256 bytes each. The next data packet is sent only after the acknowledgement for the previous packet is received. The directory of the forms is also updated.

If user chooses his option to view an article, the forum requested by the user is verified and a request is sent to the server. When the server is free it interprets the request from 'state' field in the

IPX packet and identifies that the client requests the directory of a forum and so the directory is transferred to client and displayed. This directory contains the list of articles in that forum with the id of the user who wrote it, its description and date & time when it was added to the forum. The client selects the article he wishes to view from the directory and then sends a request to the server. The server when free interprets this request and retrieves the file from the specified forum.

Limitations

1. A node is idle running the server routine.

To make the file server as a BBS server, an NLM should have been developed. But NLM development require Netware SDK, Watcom 386 C compiler. Due to the non - availability of SDK unfortunately NLM development was not possible.

2. Article names in a forum must be unique.

When there are large number of BBS users there are more chances of users thinking about the same area and giving the same title to their articles when adding their articles to the forms. But since this software requires the users to give entirely unique article name it might test the user's patience if he gives a title that is already present.

CONCLUSION

The aim of the project to facilitate the user to write articles and add them to forums and also to read articles from different forms has been implemented. The user is given a comfortable NORTON editor to type his article. The directory of each forms containing the article's name, description, sender id, makes it easier to choose his article to read.

Future Enhancements

1. The dedication of a node as a server can be eliminated by developing an NLM and loading it on the file server of the network. Once the file server becomes also the BBS server, then E-Mail facility can be easily facilitated to the BBS users, the database management can be done very efficiently.
2. The user interface can be made more friendly, in the sense, the choice of forums can be increased, more specialized forms can be created, which may cater different interest in people and also educate a lot of people with the latest technology.
3. The BBS can have an expert system that answers to the queries put forth by the BBS users. This expert system reduces the delay, in fact BBS user gets the answer for his question immediately. This will also free the human experts to an extent as they need to answer only questions that cannot be answered by the expert system.

BIBLIOGRAPHY

1. 'COMPUTER NETWORKS' ANDREW S TANENBAUM
   - PRENTICE HALL OF INDIA, 1989.

2. 'LOCAL AREA NETWORKS' GERD.E.KEISER
   - McGRAW HILL PUBLICATION, 1989.

3. 'NETWORK PROGRAMMING IN C' BARRY NANCE
   - PRENTICE HALL OF INDIA, 1994.

4. 'NETWARE PROGRAMMER'S GUIDE' JOHN T. MCCANN
   - BPB PUBLICATIONS, 1993.

5. 'PROGRAMMER'S REFERENCE MANUAL FOR IBM PERSONAL COMPUTERS' STEVEN ARMBRUST & TED FORGERON
   - GALGOTIA PUBLICATION, 1993.

6. 'MODEMS/BBS/EMAIL' ALOK SINHA & ANJALI SINHA
   - COMDEX PUBLICATIONS, 1995.

* BULLETIN BOARD SERVICE

SERVER ROUTINE

```
# include<stdio.h>
# include<dos.h>
# include<dir.h>
# include<string.h>
# include"nxt.h"
# include"nwconn.h"
# include"nwmisc.h"
# include<signal.h>
# include<io.h>
# include<process.h>
int cno,rflag,aflag,bflag;
FILE *fptr1,*fptr,*of;
char str[256],c;
int ssz=0,packetcount=0,fsize=0;
int ct.i.j;
unsigned int connection();
char who[50];
union REGS regs;
struct SREGS sregs;
struct {
 unsigned int len;
 unsigned char buftyp;
 unsigned int objtyp;
 unsigned char namlen;
 unsigned char name[47];
 } req;

struct
{
 unsigned int len;
 unsigned char no;
 unsigned cno[100];
 } rep;


# define SOCKET_TO_USE 0x6868;
# define retriesAllowed 2;
#if defined(_TINY_)||defined(_SMALL_)||defined(_MEDIUM)
# define Seg(ptr) _DS
# define Ofs(ptr) (unsigned)(ptr)
# else
# define Ofs(fp) ((unsigned)(fp))
# define Seg(fp) ((unsigned)((unsigned long)(fp)>>16))
# endif

typedef union {
                struct{
                        unsigned int Ax,Bx,Cx,Dx,Si.Di, FLAGS;
                        }F;
            struct {
                         unsigned char Al,Ah,Bl,Bh,Ch,Cl,Dl,Dh;
                         }H;
                         }Registers;
```

```
# define b "f:\\fcse\\92cse07\\bull"
# define t "f:\\fcse\\92cse07\\tec"
# define p "f:\\fcse\\92cse07\\pun"
# define r "f:\\fcse\\92cse07"

struct date dt;
struct time tm;
typedef struct {
                char tit[25];
                char id[25];
                int day;
                int mon;
                int yr;
                unsigned char hr;
                unsigned char min;
                unsigned char sec;
                char des[50];
            }msghdr;

msghdr mhd,mh;

typedef struct
            {
                IPXHeader   Hdr;
                msghdr mhead;
                char srcid[15];    /*SOURCE ID*/
                char fnam[25];     /*FILE NAME*/
                int state;         /*HEADER,DATA,ACK,DIR*/
                int pc;            /*PACKET COUNT*/
                int fsz;           /*PAC SIZE*/
                int oper;          /*OPERATION ASKED BY CLIENT*/
                int flag;          /*FLAG TO INDICATE WHETHER COMMN. POSSIBLE*/
                int forum;         /*FORUM*/
                char Data[256];    /*DATA*/
                } Packet;

ECB    ECBs,ECBr;
Packet  IPXs,IPXr;
Registers  Regs;
int timeOut,fr;
void doIPXsetup();
void doIPXsetup1();
void  doIPXwait();
void resendAbort();
void rcvPacket();
void exit(int);
int kbhit(void);
char temp[25];
int eflag,SR,fflag=0,op,count;
char usr[25],sid[25];
void rcv();
int typ;
char fname[25];

# define fam "temp1.c"
```

```
char s[30];
void send();
void send4();
void ins();

void main()
{
 while(1)
 {
  clrscr();
  system("menu.exe");
  doIPXsetup1();
  rcvPacket();
  doIPXwait();
  system("mode 80");
  system("cls");
  if(ECBr.completionCode==0)
  {
   op=IPXr.oper;
   if(op==1)
   {
    printf("\nRecieved header pac");
    switch(IPXr.forum) {
           case 1 : {
                     strcpy(s,b);
                     break;
                    }
           case 2 : {
                     strcpy(s,t);
                     break;
                    }
           case 3 : {
                     strcpy(s,p);
                     break;
                    }
                            }


  strcpy(sid,IPXr.srcid);
  strcpy(fname,IPXr.fnam);
  packetcount=IPXr.pc;
  ssz=IPXr.fsz;
  rcv();
} /*if op*/
else if(op==2)
        {
         printf("\recd oper pac");
     switch(IPXr.forum) {
          case 1 : {
                   strcpy(s,b);
                   break;
                     } /*case 1*/

          case 2 : {
                   strcpy(s,t);
                   break;
```

```
                    } /*case 2*/
            case 3 : {
                    strcpy(s,p);
                    break;
                     } /*case 3*/
                              } /* switch forum*/

           switch(IPXr.state) {
             case 1 : {
                      strcpy(sid,IPXr.srcid);
                      strcpy(fname,IPXr.fnam);
                      send();
                      break;
                     } /*case 1*/
             case 4 : {
                 strcpy(sid,IPXr.srcid);
                      strcpy(fname,"contents.c");
                      send4();
                      break;
                     } /*case 4*/
                              } /*switch state*/

          } /*if op==2*/
   } /*if ecbr. comple*/
   else
   {
    printf("did not recv header pac");
   }
  } /*while 1 stmt*/
 } /*main*/

void doIPXsetup()
  {
   BYTE status;
   WORD  socket,mySocket;
   int result;
   char myAddress[10];

   status=IPXInitialize();
   if (status != 0)
   {
    printf("\nUnable to initialize ipx interface, aborting \n");
    exit(4);
    }
    socket=SOCKET_TO_USE;
    result=IPXOpenSocket(&socket,0x00);
    if (( result) && (result != 0xff))
     {
         printf("\n failed:result of ipx open [%x],exiting to system \n\7",result);
         exit(5);
         }

         memset(&ECBs,0,sizeof(ECBs));
         memset(&IPXs,0,sizeof(IPXs));
         memset(&ECBr,0,sizeof(ECBr));
         memset(&IPXr,0,sizeof(IPXr));
```

```
        ECBs.socketNumber = socket;
        ECBs.fragmentCount = 1;
        ECBs.fragmentDescriptor[0].address = &IPXs;
        ECBs.fragmentDescriptor[0].size = 512;

        IPXs.Hdr.packetType = 4;
      GetInternetAddress(cno,&IPXs.Hdr.destination.network[0],
myAddress,&mySocket);
        memcpy(&ECBs.immediateAddress[0],myAddress,6);
        memcpy(&IPXs.Hdr.destination.socket, &socket,2);
        memcpy(&IPXs.Hdr.destination.node[0],myAddress,6);

        ECBr.socketNumber  = socket;
        ECBr.fragmentCount  = 1;
        ECBr.fragmentDescriptor[0].address  =&IPXr;
        ECBr.fragmentDescriptor[0].size   =576;
        timeOut =40;
       }


       void doIPXwait()
       {
        unsigned  int stime,time,TimeOut,resends;

        resends=time=0;
        TimeOut=timeOut;
        stime=IPXGetIntervalMarker();
        while(ECBr.inUseFlag)
        {
        time=IPXGetIntervalMarker();
        IPXRelinquishControl();
        if(((time-stime)>TimeOut)  && (ECBr.inUseFlag))
        {
         resends++;
         IPXSendPacket(&ECBs);
         stime=IPXGetIntervalMarker();
         }
         if (resends >= 100)
          {
              resendAbort();
              break;
          }
         }
        }



       void resendAbort()
       {
        printf("\n aborted,unable to communicate with anyone ");
        /*exit(11);*/
        }
```

```
               void rcvPacket()
               {
                     ECBr.fragmentCount   =1;
                     ECBr.fragmentDescriptor[0].address =&IPXr;
                     ECBr.fragmentDescriptor[0].size  =576;
                     ECBr.socketNumber   =SOCKET_TO_USE;
                     IPXListenForPacket(&ECBr);
                     }


unsigned int connection()
{
 regs.h.ah=0xe3;
 req.len=51;
 req.buftyp=0x15;
 req.objtyp=0x0100;
 req.namlen=(unsigned char)strlen(usr);
 strcpy(req.name,usr);
 rep.len=101;
 regs.x.si=FP_OFF( (void far *) &req);
 sregs.ds=FP_SEG( (void far *) &req);
 regs.x.di=FP_OFF( (void far *) &rep);
 sregs.es=FP_SEG( (void far *) &rep);

 int86x(0x21,&regs,&regs,&sregs);
 if(regs.h.al!=0)
  {
   printf("\nError Occured.");
  exit(0);
  }
 if(rep.no==0)
  {
  printf("\n%s has not logged in.",usr);
  exit(0);
  }
 regs.h.ah=0;
 regs.h.al=rep.cno[0];
 return(regs.x.ax);
}

void rcv()
{
 char d[25];
 chdir(s);
 getcurdir(0,d);
 printf("\npwd %s ",s);

 if((fptr=fopen(fname,"r"))!=NULL)
 {
  printf("\nsending ack for oper pac that file already exists");
  doIPXsetup();
  IPXs.state=3;
  IPXs.flag=1;
  IPXSendPacket(&ECBs);
  getch();
  return;
 }
 if((fptr=fopen("contents.c","r+"))==NULL)
  printf("\nerror");
 while(!feof(fptr))
 {
  fscanf(fptr,"%s\n",mhd.id);
  fscanf(fptr,"%s\n",mhd.tit);
  fscanf(fptr,"%s\n",mhd.des);
  fscanf(fptr,"%d/",mhd.day);
  fscanf(fptr,"%d/",mhd.mon);
  fscanf(fptr,"%d\n",mhd.yr);
  fscanf(fptr,"%d:",mhd.hr);
  fscanf(fptr,"%d:",mhd.min);
  fscanf(fptr,"%d\n\n",mhd.sec);
 }
 fprintf(fptr,"%s\n",IPXr.mhead.id);
 fprintf(fptr,"%s\n",IPXr.mhead.tit);
 fprintf(fptr,"%s\n",IPXr.mhead.des);
 fprintf(fptr,"%d/",IPXr.mhead.day);
 fprintf(fptr,"%d/",IPXr.mhead.mon);
 fprintf(fptr,"%d\n",IPXr.mhead.yr);
 fprintf(fptr,"%d:",IPXr.mhead.hr);
 fprintf(fptr,"%d:",IPXr.mhead.min);
 fprintf(fptr,"%d\n\n",IPXr.mhead.sec);
 fclose(fptr);

 strcpy(usr,sid);
 cno=connection();
 doIPXsetup();
 printf("\nsending ack for oper pac");
 IPXs.flag=0;
 IPXs.state=3;
 IPXSendPacket(&ECBs);
 eflag=0;
 if((fptr=fopen(fname,"w"))==NULL)
 {
  printf("\nFile name already exists");
  eflag=1;
 }
 count=1;
 while(packetcount>0)
 {
  rcvPacket();
  doIPXwait();
  if(ECBr.completionCode==0)
  {
   if(eflag==1)
   {
    doIPXsetup();
    IPXs.flag=1;
    IPXs.state=3;
    printf("\nsending ack for %d pac",count);
    IPXSendPacket(&ECBs);
    return;
   }
  if((IPXr.pc==count)&&(IPXr.state==2))
```

```
    {
     printf("\nReceived %d packet ",IPXr.pc);
     fwrite(IPXr.Data,IPXr.fsz,1,fptr);
     printf("  sss");
     doIPXsetup();
     IPXs.flag=0;
     IPXs.state=3;
     printf("\nsending ack for %d pac",count);
     IPXSendPacket(&ECBs);
     count++;
     packetcount--;
    } /*if  count stmt*/
   else
    {
     printf("\nnot correct pac");
    }
  } /*if ecbr compl stmt*/
 } /*while pac count*/
 fclose(fptr);
 chdir(r);
 getcurdir(0,d);
 printf("\npwd %s",d);
 return;
}


void send()
{
 char d[25];
 chdir(s);
 getcurdir(0,d);
 printf("\npwd %s ",s);
 strcpy(usr,sid);
 cno=connection();
 doIPXsetup();
 IPXs.flag=0;
 IPXs.state=3;
 printf("\nsending ack for oper pac");
 IPXSendPacket(&ECBs);
 aflag=1;



while(aflag)
{
 rcvPacket();
 doIPXwait();
/*  1  */

if((ECBr.completionCode==0)&&(IPXr.state==3))
{
 aflag=0;
 if ((fptr=fopen(fname,"r")) == NULL)
 {
  printf("FILE NOT AVAILABLE, mesg sent to client");
  IPXs.flag=1;
```

```
   IPXSendPacket(&ECBs);
 /*  getch(); */
   exit(0);
  } /*if fptr stmt*/
  fflush(stdin);
  rflag=1;
  while(rflag==1)
  {
   while(fsize<256)
   {
    c=fgetc(fptr);
    if(c!=EOF)
     fsize++;
    else
    {
     rflag=0;
     break;
    } /*else stmt*/
   } /*while fsize stmt*/
  j=fsize;
  packetcount++;
  ssz=fsize;
  fsize=0;
  memset(str,0,256);
 } /*while rflag stmt*/
 i=(packetcount-1)*256+ssz;
 printf("\n\n\nTotally %d bytes in %d packets",i,packetcount);
 fclose(fptr);
 IPXs.flag=0;
 IPXs.pc=packetcount;
 IPXs.state=1;
 i=packetcount;
 printf("\nSending header pac");
 IPXSendPacket(&ECBs);
 rflag=1;
 while(rflag)
 {
 rcvPacket();
 doIPXwait();

/* 2 */

if((ECBr.completionCode==0)&&(IPXr.state==3))
{
 rflag=0;
 printf("\nHeader pac sent");
 fptr=fopen(fname,"r");
 j=0;
 while(i>1)
 {
  doIPXsetup();
  fread(IPXs.Data,256,1,fptr);
  IPXs.pc=++j;
  IPXs.fsz=256;
  IPXs.state=2;
```

```
        printf("\nsending %d pac",j);
        IPXSendPacket(&ECBs);
        i--;
        bflag=1;
        while(bflag)
        {
        rcvPacket();
        doIPXwait();

        /* 3 */
        if((ECBr.completionCode==0)&&(IPXr.state==3))
        {
         printf("\n%d pac sent succesfully",j);
         bflag=0;
        } /*if ecbr compl 3 stmt*/
        else
         printf("\nnot correct pac,   only %d pacs sent",j);
   /*    getch();*/
       } /*while bflag stmt*/
      } /* while i  stmt*/
      doIPXsetup();
      fread(IPXs.Data,ssz,1,fptr);
      IPXs.pc=++j;
      IPXs.state=2;
      IPXs.fsz=ssz;
      printf("\nsending %d pac",j);
      IPXSendPacket(&ECBs);
      bflag=1;
      while(bflag)
      {
      rcvPacket();
      doIPXwait();
      if(ECBr.completionCode==0)
       {
        printf("\n%d pac sent",j);
        bflag=0;
       } /*if ecbr compl  4 stmt*/
      else
       printf("last pac not sent");
     } /*while bflag stmt*/
    } /*if ecbr compl 2 */
    else
     printf("\nwaiting ack for header pac ");
    } /*while rflag*/
   }  /*if ecbr compl 1*/
   else
    printf("\nwaiting for signal pac");
  /* getch();*/
   } /* while rflag */
   chdir(r);
   getcurdir(0,d);
   printf("\npwd %s ",s);
   return;
  }
```

```
void doIPXsetup1()
{
 BYTE status;
 WORD  socket,mySocket;
 int result;
 char myAddress[10];

 status=IPXInitialize();
 if (status != 0)
 {
  printf("\nUnable to initialize ipx interface, aborting \n");
  exit(4);
  }
  socket=SOCKET_TO_USE;
  result=IPXOpenSocket(&socket,0x00);
  if (( result) && (result != 0xff))
  {
       printf("\n failed:result of ipx open [%x],exiting to system \n\7",result);
       exit(5);
       }
       memset(&ECBs,0,sizeof(ECBs));
       memset(&IPXs,0,sizeof(IPXs));
       memset(&ECBr,0,sizeof(ECBr));
       memset(&IPXr,0,sizeof(IPXr));
       timeOut =40;
      }


void send4()
{
 char d[25];
 chdir(s);
 getcurdir(0,d);
 printf("\npwd %s ",s);
 strcpy(usr,sid);
 cno=connection();
 doIPXsetup();

 IPXs.flag=0;
 IPXs.state=3;
 printf("\nsending ack for oper pac");
 IPXSendPacket(&ECBs);
 aflag=1;
 while(aflag)
 {
  rcvPacket();
  doIPXwait();
/*  1  */

if((ECBr.completionCode==0)&&(IPXr.state==3))
{
 aflag=0;
 if ((fptr=fopen("contents.c","r")) == NULL)
 {
  printf("FILE NOT AVAILABLE, mesg sent to client");
  IPXs.flag=1;
  IPXSendPacket(&ECBs);
/*  getch(); */
  exit(0);
 } /*if fptr stmt*/
 fflush(stdin);
 fsize=0;
 while((c=fgetc(fptr))!=EOF)
  fsize++;
 ssz=fsize%256;
 packetcount=fsize/256;
 if(packetcount==0)
  packetcount=1;
 printf("\n\n\nTotally %d bytes in %d packets",fsize,packetcount);
 fclose(fptr);
 IPXs.flag=0;
 IPXs.pc=packetcount;
 IPXs.state=1;
 i=packetcount;
 printf("\nSending header pac");
 IPXSendPacket(&ECBs);
 rflag=1;
 while(rflag)
 {
 rcvPacket();
 doIPXwait();

/* 2 */

if((ECBr.completionCode==0)&&(IPXr.state==3))
{
 rflag=0;
 printf("\nHeader pac sent");
 fptr=fopen("contents.c","r");
 j=0;
 while(i>1)
 {
  doIPXsetup();
  fread(IPXs.Data,256,1,fptr);
  IPXs.pc=++j;
  IPXs.fsz=256;
  IPXs.state=2;
  printf("\nsending %d pac",j);
  IPXSendPacket(&ECBs);
  i--;
  bflag=1;
  while(bflag)
  {
  rcvPacket();
  doIPXwait();

  /* 3 */
  if((ECBr.completionCode==0)&&(IPXr.state==3))
  {
   printf("\n%d pac sent succesfully",j);
   bflag=0;
```

```
       } /*if ecbr compl 3 stmt*/
       else
        printf("\nnot correct pac,    only %d pacs sent",j);
     /*  getch();*/
       } /*while bflag stmt*/
      } /* while i  stmt*/
      doIPXsetup();
      fread(IPXs.Data,ssz,1,fptr);
      IPXs.pc=++j;
      IPXs.state=2;
      IPXs.fsz=ssz;
      printf("\nsending %d pac",j);
      IPXSendPacket(&ECBs);
      bflag=1;
      while(bflag)
      {
      rcvPacket();
      doIPXwait();
      if(ECBr.completionCode==0)
       {
        printf("\n%d pac sent",j);
        bflag=0;
      } /*if ecbr compl  4 stmt*/
     else
      printf("last pac not sent");
     } /*while bflag stmt*/
    } /*if ecbr compl 2 */
    else
     printf("\nwaiting ack for header pac ");
    } /*while rflag*/
   }  /*if ecbr compl 1*/
   else
    printf("\nwaiting for signal pac");
   /*getch();*/
   } /* while rflag */
   chdir(r);
   getcurdir(0,d);
   printf("\npwd %s ",s);
   return;
  }
```

CLIENT ROUTINE

```
# include<stdio.h>
# include<stdlib.h>
# include<dos.h>
# include<string.h>
# include"nxt.h"
# include"nwconn.h"
# include"nwmisc.h"
# include<signal.h>
# include<io.h>
# include<process.h>
# include<graphics.h>
# include<conio.h>
# include<dos.h>
# include<bios.h>

void helmen();
void frmen();
void me1();
void me2();
void fopt1();
void fopt2();
void fopt3();
void fopt4();
void hopt1();
void hopt2();
void hopt3();
void hopt4();

unsigned char connection_number;
unsigned char user_id[48];
    union REGS regs;
    struct SREGS sregs;
    struct{
            unsigned int len;
            unsigned char buffer_type;
            unsigned char connection_number;
           } request_buffer;

   struct {
            unsigned  int   len;
            unsigned  char  object_id[4];
            unsigned  char  object_type[2];
            char           object_name[48];
            char           login_time[7];
            int reserved;
           } reply_buffer;



struct node {
            char id[25];
            struct node *next;
           };
```

```
void fopt4();
void hopt1();
void hopt2();
void hopt3();
void hopt4();

unsigned char connection_number;
unsigned char user_id[48];
    union REGS regs;
    struct SREGS sregs;
    struct{
              unsigned int len;
              unsigned char buffer_type;
              unsigned char connection_number;
             } request_buffer;

    struct {
               unsigned  int   len;
               unsigned  char  object_id[4];
               unsigned  char  object_type[2];
               char           object_name[48];
               char           login_time[7];
               int reserved;
             } reply_buffer;



struct node {
              char id[25];
              struct node *next;
             };
int choice=0;
int ht,opt,place,dat,cno,rflag,aflag,bflag;
FILE *fptr1,*fptr,*of;
char str[256],c;
int ssz=0,packetcount=0,fsize=0;
int ct,i,j;
unsigned int connection();
char who[50];
union REGS regs;
struct SREGS sregs;
struct {
 unsigned int len;
 unsigned char buftyp;
 unsigned int objtyp;
 unsigned char namlen;
 unsigned char name[47];
 } req;

struct
{
 unsigned int len;
 unsigned char no;
 unsigned cno[100];
 } rep;
```

```
# define SOCKET_TO_USE 0x6868;
# define retriesAllowed 2;
#if defined(_TINY_)||defined(_SMALL_)||defined(_MEDIUM)
# define Seg(ptr) _DS
# define Ofs(ptr) (unsigned)(ptr)
# else
# define Ofs(fp) ((unsigned)(fp))
# define Seg(fp) ((unsigned)((unsigned long)(fp)>>16))
# endif

typedef union {
                    struct{
                              unsigned int Ax,Bx,Cx,Dx,Si,Di, FLAGS;
                              }F;
               struct {
                               unsigned char Al,Ah,Bl,Bh,Ch,Cl,Dl,Dh;
                               }H;
                               }Registers;

struct date dt;
struct time tm;




typedef struct {
                    char tit[25];
                    char id[25];
                    int day;
                    int mon;
                    int yr;
                    unsigned char hr;
                    unsigned char min;
                    unsigned char sec;
                    char des[50];
               }msghdr;

msghdr mhd;
typedef struct
               {
                    IPXHeader   Hdr;
                    msghdr mhead;
                    char srcid[15];    /*SOURCE ID*/
                    char fnam[25];     /*FILE NAME*/
                    int state;         /*HEADER,DATA,ACK*/
                    int pc;            /*PACKET COUNT*/
                    int fsz;           /*PAC SIZE*/
                    int oper;          /*OPERATION ASKED BY CLIENT*/
                    int flag;          /*FLAG TO INDICATE WHETHER COMMN. POSSIBLE*/
                    int forum;         /*FORUM*/
                    char Data[256];    /*DATA*/
                    } Packet;
```

```
 ECB   ECBs,ECBr;
 Packet  IPXs,IPXr;
 Registers  Regs;
 int timeOut,eflag;
 void doIPXsetup();
 void  doIPXwait();
 void resendAbort();
 void rcvPacket();
 int  getche(void);
 int getch(void);
 /*int int86 (int,Registers *,Registers *);*/
 void exit(int);
 int kbhit(void);
 char temp[25];
 int SR,fflag=0,op,count;
 char usr[25],sid[25];
 void rcv();
 int typ,fr;
 char fname[25],fam[25];

 # define bs "92cse07"


 void send();
 void rcv4();
 void ins();
 void usr_id();
 void me();
 void opt1();
 void opt2();
 void opt3();
 void opt4();

main()
{
 union REGS ir,or;
 int place,opt,eval,val;
 int gd=DETECT,gm;
 while(1)
 {
 initgraph(&gd,&gm,"c:\\tc\\bgi");
 cleardevice();
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,7);
 setbkcolor(YELLOW);
 setcolor(BLUE);
 rectangle(5,5,getmaxx()-5,getmaxy()-5);
 line(5,75,getmaxx()-5,75);
 line(5,getmaxy()-50,getmaxx()-5,getmaxy()-50);
 setcolor(LIGHTMAGENTA);
 floodfill(50,getmaxy()-25,BLUE);
 setcolor(BLUE);
 outtextxy(240,5,"BBS");
 me();
 opt1();
 place=1;
```

```
opt=0;
while(opt!=7181)
{
 opt=bioskey(0);
 switch(opt) {
  case 18432 : {
  /* UP */     switch(place) {
                    case 1 : {
                                 me();
                                 opt4();
                                 place=4;
                                 break;
                               } /*case 1*/

                    case 2 : {
                                 me();
                                 opt1();
                                 place=1;
                                 break;
                               } /*case 2*/



                    case 3 : {
                                 me();
                                 opt2();
                                 place=2;
                                 break;
                               } /*case 3*/

                    case 4 : {
                                 me();
                                 opt3();
                                 place=3;
                                 break;
                               } /*case 4*/

                                  } /*switch place*/
                  break;
                 } /*case 18432*/



 case 20480 : {
 /* DOWN */   switch(place) {
                   case 1 : {
                                me();
                                opt2();
                                place=2;
                                break;
                              } /*case 1*/

                   case 2 : {
                                me();
                                opt3();
                                place=3;
```

```
                                        break;
                                       } /*case 2*/

                              case 3 : {
                                        me();
                                        opt4();
                                        place=4;
                                        break;
                                       } /*case 3*/

                              case 4 : {
                                        me();
                                        opt1();
                                        place=1;
                                        break;
                                       } /*case 4*/

                                         } /*switch place*/
                             break;
                            } /*case 20480*/
                        } /*switch opt*/
         }
         if((place==2)||(place==3))
          frmen();
         if(place==1)
         {
          helmen();
          closegraph();
          if(ht==1)
           system("ne abbs.hlp");
          if(ht==2)
           system("ne anews.hlp");
          if(ht==3)
           system("ne view.hlp");
          continue;
         }
         closegraph();
         getdate(&dt);
         mhd.day=(int)dt.da_day;
         mhd.yr=(int)dt.da_year;
         mhd.mon=(int)dt.da_mon;
         gettime(&tm);
         mhd.hr=(int)tm.ti_hour;
         mhd.min=(int)tm.ti_min;
         mhd.sec=(int)tm.ti_sec;
         switch(place) {
          case 2 : {
                   usr_id();
                   strcpy(sid,user_id);
                   printf("\nEnter u'r article's title and give a brief decription : ");
                   scanf("%s %s",mhd.tit,mhd.des);
                   strcpy(mhd.id,sid);
                   strcpy(fname,mhd.tit);
                   fptr=fopen("temp.c","w");
                   fclose(fptr);
                   system("ne temp.c");
```

```
              eflag=1;
              send();
              while(eflag)
              {
               scanf("%s",mhd.tit);
               strcpy(fname,mhd.tit);
               send();
              }
              fcloseall();
              /*getch();*/
              break;
             } /*case 1*/
case 3 : {
             usr_id();
             strcpy(sid,user_id);
             switch(fr) {
              case 1 : {
                      fr=1;
                      rcv4();
                      flushall();
                      fcloseall();
                      if((fptr=fopen("temp.c","r"))==NULL)
                           printf("\nerrorrrrr");
                      if((fptr1=fopen("contents.c","w"))==NULL)
                           printf("\nerreor");
                      while(!feof(fptr))
                      {
                           fscanf(fptr,"%s\n",mhd.id);
                           fscanf(fptr,"%s\n",mhd.tit);
                           fscanf(fptr,"%s\n",mhd.des);
                           fscanf(fptr,"%d/",mhd.day);
                           fscanf(fptr,"%d/",mhd.mon);
                           fscanf(fptr,"%d\n",mhd.yr);
                           fscanf(fptr,"%d:",mhd.hr);
                           fscanf(fptr,"%d:",mhd.min);
                           fscanf(fptr,"%d\n\n",mhd.sec);

                           fprintf(fptr1,"Id          : %s\n",mhd.id);
                           fprintf(fptr1,"Title       : %s\n",mhd.tit);
                           fprintf(fptr1,"Description : %s\n",mhd.des);
                           fprintf(fptr1,"Date        : %d/",mhd.day);
                           fprintf(fptr1,"%d/",mhd.mon);
                           fprintf(fptr1,"%d\n",mhd.yr);
                           fprintf(fptr1,"Time        :%d:",mhd.hr);
                           fprintf(fptr1,"%d:",mhd.min);
                           fprintf(fptr1,"%d\n\n",mhd.sec);
                      }
                      fclose(fptr);
                      fclose(fptr1);
                      fcloseall();
                      flushall();
                      system("ne contents.c");
                      printf("\nEnter the article's name");
                      scanf("%s",fname);
                      rcv();
                      fflush(fptr);
```

```
                 fflush(fptr1);
                 fcloseall();
                 flushall();
                 system("ne temp.c");
                 break;
                } /* case 1*/
   case 2 : {
                 fr=2;
                 rcv4();
                fcloseall();
                 flushall();
      if((fptr=fopen("temp.c","r"))==NULL)
                     printf("\nerrorrrrr");
                 if((fptr1=fopen("contents.c","w"))==NULL)
                     printf("\nerreor");
                 while(!feof(fptr))

                 {
                     fscanf(fptr,"%s\n",mhd.id);
                     fscanf(fptr,"%s\n",mhd.tit);
                     fscanf(fptr,"%s\n",mhd.des);
                     fscanf(fptr,"%d/",mhd.day);
                     fscanf(fptr,"%d/",mhd.mon);
                     fscanf(fptr,"%d\n",mhd.yr);
                     fscanf(fptr,"%d:",mhd.hr);
                     fscanf(fptr,"%d:",mhd.min);
                     fscanf(fptr,"%d\n\n",mhd.sec);

                     fprintf(fptr1,"Id %s\n",mhd.id);
                     fprintf(fptr1,"Title %s\n",mhd.tit);
                     fprintf(fptr1,"Des %s\n",mhd.des);
                     fprintf(fptr1,"Date %d/",mhd.day);
                     fprintf(fptr1,"%d/",mhd.mon);
                     fprintf(fptr1,"%d\n",mhd.yr);
                     fprintf(fptr1,"Time %d:",mhd.hr);
                     fprintf(fptr1,"%d:",mhd.min);
                     fprintf(fptr1,"%d\n\n",mhd.sec);
                }
                fclose(fptr);
                fclose(fptr1);
                system("ne contents.c");
                printf("\nEnter the article's name");
                scanf("%s",fname);
                rcv();
                fflush(fptr);
                fflush(fptr1);
               fcloseall();
                flushall();
                system("ne temp.c");
                break;
               }
 case 3 : {
                fr=3;
                rcv4();
               fcloseall();
                flushall();
                if((fptr=fopen("temp.c","r"))==NULL)
                        printf("\nerrorrrrr");
                     if((fptr1=fopen("contents.c","w"))==NULL)
                        printf("\nerreor");
                     while(!feof(fptr))
                     {
                        fscanf(fptr,"%s\n",mhd.id);
                        fscanf(fptr,"%s\n",mhd.tit);
                        fscanf(fptr,"%s\n",mhd.des);
                        fscanf(fptr,"%d/",mhd.day);
                        fscanf(fptr,"%d/",mhd.mon);
                        fscanf(fptr,"%d\n",mhd.yr);
                        fscanf(fptr,"%d:",mhd.hr);
                        fscanf(fptr,"%d:",mhd.min);
                        fscanf(fptr,"%d\n\n",mhd.sec);

                        fprintf(fptr1,"Id %s\n",mhd.id);
                        fprintf(fptr1,"Title %s\n",mhd.tit);
                        fprintf(fptr1,"Des %s\n",mhd.des);
                        fprintf(fptr1,"Date %d/",mhd.day);
                        fprintf(fptr1,"%d/",mhd.mon);
                        fprintf(fptr1,"%d\n",mhd.yr);
                        fprintf(fptr1,"Time %d:",mhd.hr);
                        fprintf(fptr1,"%d:",mhd.min);
                        fprintf(fptr1,"%d\n\n",mhd.sec);
                     }
                     fclose(fptr);
                     fclose(fptr1);
                     system("ne contents.c");
                     printf("\nEnter the article's name");
                     scanf("%s",fname);
                     rcv();
                     fflush(fptr);
                     fflush(fptr1);
                    fcloseall();
                     flushall();
                     system("ne temp.c");
                     break;
                    }
                    } /*switch case*/
           /*getch();*/
           break;
          } /*case 2*/



  case 4 : {
           exit(1);
          } /* case 3*/
                }/*switch choice*/

 } /*while stmt*/
} /*main*/
```

```
void doIPXsetup()
{
  BYTE status;
  WORD  socket,mySocket;
  int result;
  char myAddress[10];

  status=IPXInitialize();
  if (status != 0)
  {
    printf("\nUnable to initialize ipx interface, aborting \n");
    exit(4);
    }

    socket=SOCKET_TO_USE;
    result=IPXOpenSocket(&socket,0x00);
    if (( result) && (result != 0xff))
    {
          printf("\n failed:result of ipx open [%x],exiting to system \n\7",result);
          exit(5);
          }
           memset(&ECBs,0,sizeof(ECBs));
          memset(&IPXs,0,sizeof(IPXs));
          memset(&ECBr,0,sizeof(ECBr));
          memset(&IPXr,0,sizeof(IPXr));

          ECBs.socketNumber = socket;
          ECBs.fragmentCount = 1;
          ECBs.fragmentDescriptor[0].address = &IPXs;
          ECBs.fragmentDescriptor[0].size = 512;

          IPXs.Hdr.packetType = 4;
      GetInternetAddress(cno,&IPXs.Hdr.destination.network[0],
myAddress,&mySocket);

          memcpy(&ECBs.immediateAddress[0],myAddress,6);
          memcpy(&IPXs.Hdr.destination.socket, &socket,2);
          memcpy(&IPXs.Hdr.destination.node[0],myAddress,6);

          ECBr.socketNumber   = socket;
          ECBr.fragmentCount  = 1;
          ECBr.fragmentDescriptor[0].address =&IPXr;
          ECBr.fragmentDescriptor[0].size    =576;
          timeOut =40;
         }

          void doIPXwait()
          {
            unsigned  int stime,time,TimeOut,resends;

            resends=time=0;
            TimeOut=timeOut;
            stime=IPXGetIntervalMarker();

            while(ECBr.inUseFlag)
            {
```

```
            time=IPXGetIntervalMarker();
            IPXRelinquishControl();
            if(((time-stime)>TimeOut)  && (ECBr.inUseFlag))
            {
             resends++;
             IPXSendPacket(&ECBs);
             stime=IPXGetIntervalMarker();
             }



             if (resends >= 110)
              {
               resendAbort();
               return;
              }
             }
             }


             void resendAbort()
             {
              printf("\n aborted,unable to communicate with anyone ");
              /*exit(11);*/
              }


              void rcvPacket()
              {
                    ECBr.fragmentCount   =1;
                    ECBr.fragmentDescriptor[0].address =&IPXr;
                    ECBr.fragmentDescriptor[0].size  =576;
                    ECBr.socketNumber   =SOCKET_TO_USE;
                    IPXListenForPacket( &ECBr);
                    }


unsigned int connection()
{
 regs.h.ah=0xe3;
 req.len=51;
 req.buftyp=0x15;
 req.objtyp=0x0100;
 req.namlen=(unsigned char)strlen(usr);
 strcpy(req.name,usr);
 rep.len=101;
 regs.x.si=FP_OFF( (void far *) &req);
 sregs.ds=FP_SEG( (void far *) &req);
 regs.x.di=FP_OFF( (void far *) &rep);
 sregs.es=FP_SEG( (void far *) &rep);

 int86x(0x21,&regs,&regs,&sregs);
 if(regs.h.al!=0)
  {
   printf("\nError Occured.");
  exit(0);
  }
 if(rep.no==0)
  {
  printf("\n%s has not logged in.",usr);
  exit(0);
  }
 regs.h.ah=0;


 regs.h.al=rep.cno[0];
 return(regs.x.ax);
}


void rcv()
{
 strcpy(usr,bs);
 cno=connection();
 doIPXsetup();
 IPXs.state=1;
 IPXs.forum=fr;
 strcpy(IPXs.srcid,sid);
 strcpy(IPXs.fnam,fname);
 IPXs.oper=2;
 printf("\nsending oper pac");
 IPXSendPacket(&ECBs);
 printf("\nffffffffff");
 rflag=1;
 while(rflag)
 {
  rcvPacket();
 doIPXwait();
 if((ECBr.completionCode==0)&&(IPXr.state==3)&&(strcmpi(IPXr.srcid,sid)))
 {
  rflag=0;
  printf("\noper pac sent");
  count=1;
  doIPXsetup();
  IPXs.flag=0;
  IPXs.forum=fr;
  IPXs.state=3;
 printf("\nsending start signal");
 IPXSendPacket(&ECBs);
 aflag=1;
 while(aflag)
 {
  rcvPacket();
  doIPXwait();
 if((ECBr.completionCode==0)&&(IPXr.state==1)&&(strcmpi(IPXr.srcid,sid)))
 {
  aflag=0;
  if(IPXr.flag==1)
  {
   printf("\nfile not available");
   getch();
  }
```

```
      printf("\n\nHeader pac recd.");
      packetcount=IPXr.pc;
      doIPXsetup();
      printf("\nSending ack for header");
      IPXs.state=3;
      IPXSendPacket(&ECBs);
      fptr=fopen("temp.c","w");
      while(packetcount>0)
      {
       bflag=1;
       while(bflag)
       {
        rcvPacket();
        doIPXwait();
        if((ECBr.completionCode==0)&&(IPXr.state==2))
        {
             if((IPXr.pc==count)&&(IPXr.state==2))
             {
              bflag=0;
              printf("\nReceived %d packet ",IPXr.pc);
              fwrite(IPXr.Data,IPXr.fsz,1,fptr);
              printf("  sss");
              doIPXsetup();
              IPXs.flag=0;
              IPXs.state=3;
              printf("\nSending ack for %d pac",count);
              IPXSendPacket(&ECBs);
              count++;
          packetcount--;
             } /*if  count stmt*/
             else
             {
              printf("\nrecd. not correct pac");
             }
       } /*if ecbr compl 3*/
      } /*while bflag*/
     } /*while pac count*/
    } /* if ecbr compl 2*/
    else
     printf("\nwaiting for header pac");
   } /*while aflag*/
  } /*ecbr compl*/
  else
   printf("\noper ppac not sent");
 } /*while rflag*/
 fclose(fptr);
 return;
}

void send()
{
 strcpy(usr,bs);
 cno=connection();

 doIPXsetup();
 if ((fptr=fopen("temp.c","r")) == NULL)
```

```
  {
   printf("FILE NOT AVAILABLE");
   getch();
   exit(0);
  } /*if fptr stmt*/
  fflush(stdin);
  rflag=1;
  while(rflag==1)
  {
   while(fsize<256)
   {
    c=fgetc(fptr);
    if(c!=EOF)
     fsize++;
    else
    {
     rflag=0;
     break;
    } /*else stmt*/
   } /*while fsize stmt*/
   j=fsize;
   packetcount++;
   ssz=fsize;
   fsize=0;
   memset(str,0,256);
  } /*while rflag stmt*/
  i=(packetcount-1)*256+ssz;
  printf("\nTotally %d bytes in %d packets",i,packetcount);
  fclose(fptr);
  IPXs.forum=fr;
  IPXs.flag=0;
  IPXs.state=1;
  IPXs.mhead=mhd;
  printf("\n%d %d",IPXs.mhead.day,IPXs.mhead.mon);
  strcpy(IPXs.srcid,sid);
  strcpy(IPXs.fnam,fname);
  IPXs.pc=packetcount;
  IPXs.oper=1;
  IPXs.fsz=ssz;
  i=packetcount;
  IPXSendPacket(&ECBs);
  rflag=1;
  while(rflag)
  {
   rcvPacket();
   doIPXwait();
   if((ECBr.completionCode==0)&&(IPXr.state==3))
   {
    if(IPXr.flag==1)
    {
     printf("\ngive a different title for the article");
     eflag=1;
     return;
    }
    else
     eflag=0;
```

```
 printf("\nheader pac sent succesfully");
 rflag=0;
 fptr=fopen("temp.c","r");
 j=0;
 while(i>1)
 {
  doIPXsetup();
  fread(IPXs.Data,256,1,fptr);
  IPXs.pc=++j;
  IPXs.fsz=256;
  IPXs.state=2;
  IPXSendPacket(&ECBs);
  i--;
  aflag=1;
  while(aflag)
  {
   rcvPacket();
   doIPXwait();
   if((ECBr.completionCode==0)&&(IPXr.state==3))
   {
    aflag=0;
    printf("\n%d pac sent succesfully",j);
   } /*if ecbr compl stmt*/
   else
   {
    printf("\n not correct pac,  only %d pacs sent",j);
   } /* else ecbr compl*/
/*    getch();*/
  } /*while aflag*/
 } /*while i*/
 doIPXsetup();
 fread(IPXs.Data,ssz,1,fptr);
 IPXs.pc=++j;
 IPXs.fsz=ssz;
 IPXs.state=2;
 IPXSendPacket(&ECBs);
 aflag=1;
 while(aflag)
 {
  rcvPacket();
  doIPXwait();
  if((ECBr.completionCode==0)&&(IPXr.state==3))
  {
   aflag=0;
   printf("\n%d pac sent successfully",j);
  } /*if ecbr compl stmt*/
  else
   printf("last pac not sent");
 } /*while aflag*/
 } /*if ecbr compl 1 */
 else
  printf("\nwaiting ack for header");
} /*while rflag */
return;
}
```

```
void usr_id()
{
  connection_number=GetConnectionNumber();
  regs.h.ah = 0xe3;
  request_buffer.len =2;
  request_buffer.buffer_type =0x16;
  request_buffer.connection_number = connection_number;
  reply_buffer.len = 63;
  regs.x.si =FP_OFF((void far *) &request_buffer);
  sregs.ds  =FP_SEG((void far *) &request_buffer);
  regs.x.di =FP_OFF((void far *) &reply_buffer);
  sregs.es  =FP_SEG((void far *) &reply_buffer);
  int86x(0x21,&regs,&regs,&sregs);
  strncpy(user_id,reply_buffer.object_name,48);
  return;
}

void rcv4()
{
 strcpy(usr,bs);
 cno=connection();
 doIPXsetup();
 IPXs.state=4;
 IPXs.oper=2;
 IPXs.forum=fr;
 strcpy(IPXs.srcid,sid);
 IPXs.oper=2;
 printf("\nsending oper pac");
 IPXSendPacket(&ECBs);
 printf("\nfffffffffff");
 rflag=1;
 while(rflag)
 {
   rcvPacket();
  doIPXwait();
  if((ECBr.completionCode==0)&&(IPXr.state==3)&&(strcmpi(IPXr.srcid,sid)))
  {
   rflag=0;
   printf("\noper pac sent");
   count=1;
   doIPXsetup();
   IPXs.flag=0;
   IPXs.forum=fr;
   IPXs.state=3;
   printf("\nsending start signal");
   IPXSendPacket(&ECBs);
   aflag=1;
   while(aflag)
   {
    rcvPacket();
    doIPXwait();

    if((ECBr.completionCode==0)&&(IPXr.state==1)&&(strcmpi(IPXr.srcid,sic)))
{
 aflag=0;
 if(IPXr.flag==1)
 {
  printf("\nfile not available");
  getch();
 }
 printf("\n\nHeader pac recd.");
 packetcount=IPXr.pc;
 doIPXsetup();
 printf("\nSending ack for header");
 IPXs.state=3;
 IPXSendPacket(&ECBs);
 if((fptr=fopen("temp.c","w"))==NULL)
  printf("\nerrrrrrrrorrrrr");
 while(packetcount>0)
 {
  bflag=1;
  while(bflag)
  {
   rcvPacket();
   doIPXwait();
   if((ECBr.completionCode==0)&&(IPXr.state==2))
   {
        if((IPXr.pc==count)&&(IPXr.state==2))
        {
         bflag=0;
         printf("\nReceived %d packet ",IPXr.pc);
         fwrite(IPXr.Data,IPXr.fsz,1,fptr);
         printf("  sss");
         doIPXsetup();
         IPXs.flag=0;
         IPXs.state=3;
         printf("\nSending ack for %d pac".count);
         IPXSendPacket(&ECBs);
         count++;
      packetcount--;
        } /*if  count stmt*/
        else
        {
         printf("\nrecd. not correct pac");
        }
     } /*if ecbr compl 3*/
    } /*while bflag*/
   } /*while pac count*/
  } /* if ecbr compl 2*/
  else
   printf("\nwaiting for header pac");
 } /*while aflag*/
} /*ecbr compl*/
else
 printf("\noper ppac not sent");
} /*while rflag*/
fclose(fptr);
return;
}
```

```
outtextxy(255,150,"OPTIONS");
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(230,190,"HELP");
outtextxy(230,230,"ADD NEWS");
outtextxy(230,270,"VIEW THE BBS");
outtextxy(230,310,"EXIT");
}



void me1()
{
 for(i=0,j=0;i<50,j<100;i++,j++)
 {
  setcolor(GREEN);
  rectangle(320-i,240-j,320+i,240+j);
 }
 setcolor(RED);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
 outtextxy(255,150," HELP  ");
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
 outtextxy(230,190,"ABOUT BBS");
 outtextxy(230,230,"TO ADD NEWS");
 outtextxy(230,270,"TO VIEW BBS");
 outtextxy(230,310,"EXIT");
 }



 void me2()
 {
  for(i=0,j=0;i<50,j<100;i++,j++)
  {
   setcolor(GREEN);
   rectangle(320-i,240-j,320+i,240+j);
  }
  setcolor(RED);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
  outtextxy(255,150," FORUM");
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
  outtextxy(230,190,"BULLETIN");
  outtextxy(230,230,"TECHNICAL");
  outtextxy(230,270,"FUN");
  }




  /*HELP MENU ROUTINE*/
  void helmen()
  {

   me1();
```

```
void opt1()
{
 setcolor(BLACK);
 rectangle(225,185,415,220);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,200,BLACK);
 setcolor(RED);
 outtextxy(230,190,"HELP");
}

void opt2()
{
 setcolor(BLACK);
 rectangle(225,225,415,260);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,240,BLACK);
 setcolor(RED);
 outtextxy(230,230,"ADD NEWS");
}

void opt3()
{
 setcolor(BLACK);
 rectangle(225,265,415,300);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,275,BLACK);
 setcolor(RED);
 outtextxy(230,270,"VIEW THE BBS");
}

void opt4()
{
 setcolor(BLACK);
 rectangle(225,305,415,335);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,315,BLACK);
 setcolor(RED);
 outtextxy(230,310,"EXIT");
}



void me()
{
 for(i=0,j=0;i<50,j<100;i++,j++)
 {
  setcolor(GREEN);
  rectangle(320-i,240-j,320+i,240+j);
 }
 setcolor(RED);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
```

```
hopt1();
ht=1;
opt=0;
while(opt!=7181)
{
 opt=bioskey(0);
 switch(opt) {
  case 18432 : {
   /* UP */     switch(ht) {
                      case 1 : {
                                     me1();
                                     hopt4();
                                     ht=4;
                                     break;
                                    } /*case 1*/

                       case 2 : {
                                     me1();
                                     hopt1();
                                     ht=1;
                                     break;
                                    } /*case 2*/

                       case 3 : {
                                     me1();
                                     hopt2();
                                     ht=2;
                                     break;
                                    } /*case 3*/

                        case 4 : {
                                     me1();
                                     hopt3();
                                     ht=3;
                                     break;
                                    } /*case 4*/

                                        } /*switch place*/
                       break;
                      } /*case 18432*/



    case 20480 : {
    /* DOWN */   switch(ht) {
                        case 1 : {
                                      me1();
                                      hopt2();
                                      ht=2;
                                      break;
                                     } /*case 1*/

                         case 2 : {
                                      me1();
                                      hopt3();
                                      ht=3;
```

```
                              break;
                             } /*case 2*/

                   case 3 : {
                              me1();
                              hopt4();
                              ht=4;
                              break;
                             } /*case 3*/

                   case 4 : {
                              me1();
                              hopt1();
                              ht=1;
                              break;
                             } /*case 4*/

                               } /*switch place*/
                  break;
                 } /*case 20480*/
               } /*switch opt*/
 }
}




/*FORUM MENU ROUTINE*/
void frmen()
{

 me2();
 fopt1();
 fr=1;
 opt=0;
 while(opt!=7181)
 {
  opt=bioskey(0);
  switch(opt) {
   case 18432 : {
    /* UP */    switch(fr) {
                   case 1 : {
                              me2();
                              fopt3();
                              fr=3;
                              break;
                             } /*case 1*/

                   case 2 : {
                              me2();
                              fopt1();
                              fr=1;
                              break;
                             } /*case 2*/

                   case 3 : {
```

```
                              me2();
                              fopt2();
                              fr=2;
                              break;
                             } /*case 3*/

                                } /*switch place*/
                  break;
                 } /*case 18432*/



  case 20480 : {
   /* DOWN */   switch(fr) {
                  case 1 : {
                              me2();
                              fopt2();
                              fr=2;
                              break;
                             } /*case 1*/

                  case 2 : {
                              me2();
                              fopt3();
                              fr=3;
                              break;
                             } /*case 2*/

                  case 3 : {
                              me2();
                              fopt1();
                              fr=1;
                              break;
                             } /*case 3*/

                                } /*switch place*/
                  break;
                 } /*case 20480*/
             } /*switch opt*/
 }
}


void hopt1()
{
 setcolor(BLACK);
 rectangle(225,185,415,220);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,200,BLACK);
 setcolor(RED);
 outtextxy(230,190,"ABOUT BBS");
}

void hopt2()
{
 setcolor(BLACK);
 rectangle(225,225,415,260);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,240,BLACK);
 setcolor(RED);
 outtextxy(230,230,"TO ADD NEWS");
}

void hopt3()
{
 setcolor(BLACK);
 rectangle(225,265,415,300);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,275,BLACK);
 setcolor(RED);
 outtextxy(230,270,"TO VIEW BBS");
}

void hopt4()
{
 setcolor(BLACK);
 rectangle(225,305,415,335);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,315,BLACK);
 setcolor(RED);
 outtextxy(230,310,"EXIT");
}


void fopt1()
{
 setcolor(BLACK);
 rectangle(225,185,415,220);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,200,BLACK);
 setcolor(RED);
 outtextxy(230,190,"BULLETIN");
}

void fopt2()
{
 setcolor(BLACK);
 rectangle(225,225,415,260);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,240,BLACK);
 setcolor(RED);
 outtextxy(230,230,"TECHNICAL");
}

void fopt3()
{
 setcolor(BLACK);
 rectangle(225,265,415,300);
 setfillstyle(SLASH_FILL,BLACK);
 floodfill(240,275,BLACK);
 setcolor(RED);
 outtextxy(230,270,"FUN");
```