الخلاصة

مع النمو السريع والتطور لشبكات المعلومات والانترنت، اصبح من المهم ايجاد طرق لتطوير هذه الشبكات، من خلال إضافة تطبيقات جديدة اليها. التطبيق المقترح في هذه الأطروحة هو توفير خدمة نقل الملف النصي والذي تم تطبيقه على شبكة محلية (LAN) وباستخدام نموذج الـ (TCP/IP).

ان اساس عمل التطبيق في هذا المشروع يعتمد على قيام موفر الخدمة بتوفير الخدمة الى طالب الخدمة والتي تتمثل بتحميل الملفات من والى طالب الخدمة. ان التطبيق يوفر حماية للبيانات المنقولة من خلال تشفيرها باستخدام نظام (RSA)، بالاضافة الى انه يستخدم طريقة الـ (LZW) لضغط الملفات قبل ارسالها لتقليل الوقت المستغرق لنقلها هذا وان عملية الضغط تعتبر اول مستوى من مستويات الامن للبيانات المنقولة.

ان نموذج الـ (TCP/IP) المعتمد في هذا المشروع يختلف عن بقية النماذج مثل الـ (IPX/SPX) و (DDP) التي تحتاج الى التوافق عند العمل، بانه لايحتاج الى حالة التوافق مما يكسبه فعالية اكثر عند الاستخدام.

إن هذا البحث المقترح يقوم بدراسة نموذج الـ (TCP/IP) والبحث في ايجاد امكانية اضافة تطبيقات جديدة اليه من خلال استخدام أحدى وحدات ربط التطبيقات التي تدعى الـ(sockets). إن التطبيق الذي تم اضافته في هذا المشروع هو ’’بروتوكول نقل الملفات النصية‘‘ الى نموذج الـ (TCP/IP).

النظام قد تم بناءهُ باستخدام اللغة البرمجية ’’Jbuilder‘‘ الاصدار الرابع، ونفذ في بيئة نظام التشغيل ’’Windows XP‘‘ على حاسبات شخصية تم ربطها في شبكة محلية ’’LAN‘‘ على شكل نجمة.

Abstract

With the recent growth and development of the Internet and companies intranets, it has become important to look for ways of developing these network, by developing new applications or protocols to the network reference models.

Some protocols, such as TCP/IP (Transmission Control Protocol / Internet Protocol), are vendor–neutral. Others, such as Novell IPX/SPX (Internetwork Packet Exchange / Sequenced Packet Exchange) and Apple Delivery Datagram Protocol (DDP), are tied to specific vendors.

In this thesis, the proposed application is a text file transfer service that operates on a LAN (Local Area Network) and over TCP/IP networks. The proposed application is basically a client/server protocol in which a system running the server accepts commands from a system running a client. The service allows users to send commands to the server for uploading and downloading files. The application uses the RSA encryption system to provide good security for data and LZW (Lempel Ziv Welch) compression method to reduce the transfer time, which is considered level one of security.

The proposed research study the TCP/IP protocols and improve a way of inserting new application (protocol) for the TCP/IP reference model by using the socket API (Application Programming Interface). The project adds a text file transfer protocol to the TCP/IP.

The system is build by using Jbuilder version 4 programming language, and it is implemented in Window XP operating system environment on PCs linked by LAN as a star topology.

Key Words

Application Programming Interface

Automatic Repeat request

American Standard Code for Information Interchange

Berkeley Software Distributed

Delivery Datagram Protocol

Electronic Mail System

Fiber Distributed Data Interface

File Transfer Protocol

Greatest Common Divisor

Interface Control Information

Internet Protocol

Internetwork Packet Exchange Sequenced Packet Exchange

Local Area Network

Lempel Ziv Welch

Metropolitan Area Network

Management Information Base

Network Interface Card

Personal Computer

Protocol Data Unit

Request For Comment

Remote Procedure Call

Rivest Shamir Adleman

Service Data Unit

Simple Mail Transfer Protocol

Simple Network Management Protocol

Transmission Control Protocol

Transmission Control Protocol Internet Protocol

Transport Layer Interface

Transport Protocol Data Unit

User Datagram Protocol

Wide Area Network

1-1 Introduction

FTP stands for File Transfer Protocol. It allows people to access files stored on the FTP server and transfer them to their own computers. This system allows large files of any type-notably executable and compressed files to be transferred directly from the Web interface. This is not dramatically different from existing LAN transfer systems, except that it can be built into the Web. In fact, it can be invisible to the user, who simply clicks on the name of the file on a Web page, and the FTP function is launched behind the scenes.

FTP is one of the oldest methods for moving files from one computer to another in the Internet. Before the World Wide Web (WWW) became famous, the most traffic was generated by this protocol. Today FTP is not so well known anymore. Instead of putting huge files on a ftp server, people attach the files to an e-mail and send them this way to an addressee. But that is not the way it should be done - a reason to teach people a little bit about FTP. There are a lot of fancy FTP programs available, making the use of FTP easier. And also Windows supports FTP file up- and download using the Internet Explorer. That way, FTP is nothing more than moving files from one folder in another.

FTP supports both interactive and batch mode. The most users work interactive with FTP. They are running a FTP-client and establish a connection to a server to transfer files. Some applications activate FTP automatically without inputs from the user. In this case the user will only be informed if the operation was successful or not. When a user uses FTP in the interactive mode he works either with the command line of FTP or an application with a nice user graphical interface. This way the user is allowed to drag and drop files instead of using cryptically commands, but internally the application is also using the FTP commands.

FTP is an Internet file transfer service that operates on the Internet and over TCP/IP networks. FTP is basically a client/server protocol in which a system running the FTP server accepts commands from a system running an FTP client. The service allows users to send commands to the server for uploading and downloading files. FTP operates among heterogeneous systems and allows users on one system to interact with another type of system without regard for the operating systems in place, as long as the network protocol is TCP/IP. **[CIS 99]**

1-2 Aim of thesis

The aim of thesis is to design and implement a file management protocol for text files and apply it as one application of TCP/IP reference model.

1-3 Literature Survey

- ❖ In 1990, Sam and Steve [SAM 90] build reference model which identify the high-level abstractions that underlie modern storage systems. Their model provides a common terminology and set of concepts to allow existing systems to be examined and new systems to be discussed and built.

- In 1992, Yassen [YAS 92] designed and build models for text compression, to reduce the amount of space needed to store files. Considerable part of his research was dedicated towards finding a conclusive single model to cover all types of text files, programs, and Arabic text.

- In 2000 Esraa [ESR 00] established three authentication protocols by HMAC (Key Hashing for Message Authentication Code system) system and RSA public key cryptosystem. To achieve maximum authentication of these protocols, HMAC system is used in the first stage, which depends on a one-way hash function characteristic. To increase authentication, RSA public key cryptosystem is used in the second stage, which depends on the trap–door one–way function. The use of one–way hash functions in a HMAC system with RSA public key cryptosystem is to compress any large file in a secure manner before encrypted with a private key.

- In 2002 Bassam [BAS 02] have designed and implemented an end-to-end cryptography software system that works on-line and under Windows operating system for Ethernet networks, and applied an encryption algorithm to prove the system capability to encrypt/decrypt packets on-line without crashing the Windows operating system or effecting network activities. Also, he have designed and implemented a hardware encoder/decoder card to implement the encryption algorithm by hardware and interface this card to end-to-end cryptography software system.

1-4 Thesis Outline

The contents of individual chapters of this thesis are briefly reviewed.

- ***Chapter two***: covers the theoretical basis of networking, RSA public key cryptosystem, and LZW dictionary compression method.
- ***Chapter three***: presents the proposed system architecture, and the algorithms that used to implement this system.
- ***Chapter four***: presents the user interface for the designate system.
- ***Chapter five***: introduces conclusions on this work, with recommendation for future work.

2-1 Introduction

For networks to communicate efficiently, they require a standard, or protocol. In a general sense, a protocol is an agreed-upon way to communicate. Network protocols are established rules that enable data to flow from one Network Interface Card (NIC) to another. Unless the programmers understand the specific rules applied to network communications, he/she will not be able to administer a network efficiently. **[PAR 99]**

2-2 The Need For Protocol Architecture

When computers, terminals, and/or other data processing devices exchange data, the procedures involved are the following:

- The source system must either activate the direct data communication network of the identity of the desired destination system.
- The source system must ascertain that the destination system is prepared to receive data.
- The file transfer application on the source system must ascertain that the file management program on the destination system is prepared to accept and store the file for this particular user.

If the file formats used on the two systems are incompatible, one or the other system must perform a format translation function. **[AND 89]**

In protocol architecture, the modules are arranged in a vertical stack where complex programs can only be written and verified if they are decomposed into *manageable* models. A module is a block of code which is specified by the action it performs and by the way it interacts with other modules. The module can be constructed independently of the rest of the software, as long as it respects the interaction and functions. **[JEA 91]**

2-3 Layered Network Architecture

To reduce their design complexity, most networks are organized as a series of layers or levels, each one built upon its predecessor, where each layer offers certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. **[AND 89]**

Typically, a service of layer N is executed by peer protocol entities in different nodes of a communication network. The messages exchanged by peer protocol entities of layer N are called layer N Protocol Data Units (N-PDUs). The messages exchanged by services of layer N are also called layer N Services Data Units (N-SDUs). Thus, the N+1-PDU is an N-SDU.**[JEA 91]**

In figure 2.1,when protocol entity of layer N+1 in A wants to send an N+1-PDU to a peer protocol entity in B. The protocol entity uses a *service primitive* of layer N called an N.request. This request is placed by sending some Interface Control Information (ICI) to layer N. The ICI specifies the request type as well as some parameters, such as the addresses of the nodes, and the protocol entities inside the nodes and, possibly, a description of the desired quality of the information transfer service. The service provided by layer N, called the N_SERVICE, eventually sends an N-indication to the protocol entity in layer N+1 of B. At that time, the protocol entity normally receives the N+1-PDU. The protocol entity in B later sends an N.response as a reply to the N.indication. The N.response is another N-SERVICE primitive which contains parameters that describe the reply to the N.request. This response eventually gets back to the protocol entities in layer N+1 of A as N.confirm. **[JEA 91]**

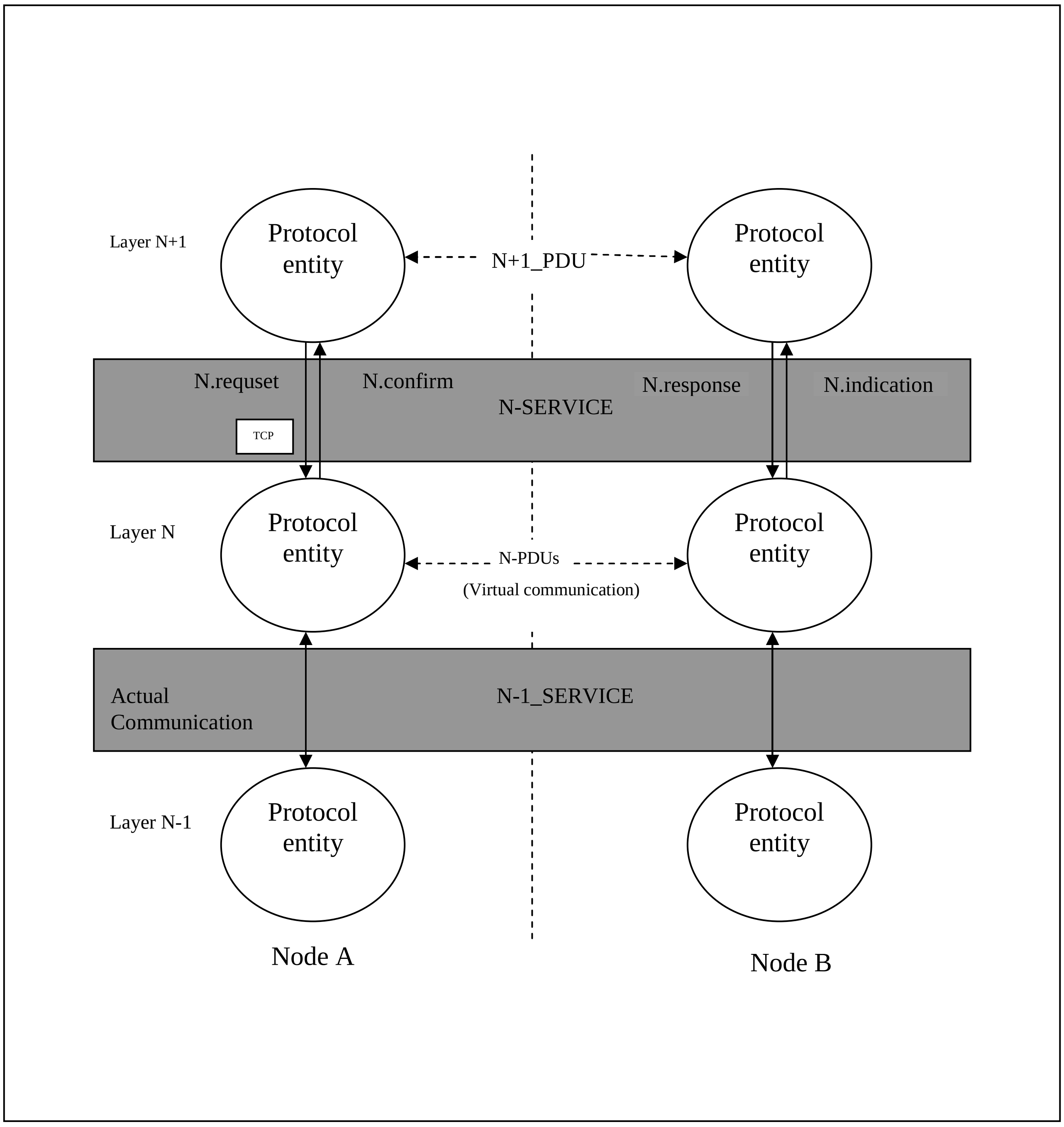


Figure 2.1-Layered Architecture

2-4 Design Issues For The Layers

Some of the key design issues that occur in computer networking are presented in several layers, below some important of them:

2-4-1 Connection establishing and terminating

Every layer must have a mechanism for connection establishment. Since a network normally has many computers, some of which have multiple processes, a means is needed for a process on one machine to specify with whom it wants to establish a connection. As a sequence of having multiple destinations, some form of addressing is needed in order to specify a specific destination. Closely related to the mechanism for establishing connections across the network is the mechanism for terminating them once they are no longer needed. **[AND 89]**

2-4-2 Rules for data transfer

When data is transmitted between two pieces of equipment, three analogous modes of operation can be used:

- Simplex: This is used when data is transmitted in one direction only
- Half duplex: This used when the two interconnected devices wish to exchange information (data) alternately.

Duplex: This is also referred to as full-duplex and is used when data is to be exchanged between the two connected devices in both directions simultaneously. **[FRE 96]**

2-4-3 Error control

Error control is an important issue because physical communication circuits are not perfect. **[AND 89]**

When a computer is transferring blocks of characters (frames) across a serial data link to another computer, the program in the receiving computer control procedure automatically without any intervention from the user. Typically, the receiving computer checks the received frame for possible transmission errors and then returns a short message (frame) either to acknowledge its correct receipt or to request that another copy of the frame is sent. This type of error control is known as Automatic Repeat request (ARQ). **[FRE 96]**

2-4-4 Multiplexing

Multiplexing is the transmission of different flows of information on the same physical link; flow information means a sequence of packets or a bit stream sent by one user to another user. One simple way to implement multiplexing of sequences of packets, called *statical multiplexing*, is to store in the same buffer all the packets that need to be transmitted over a given line. **[AND 89]**

2-4-5 Routing

Routing is the process of selecting a path over which to send packets. Routing includes two types:

- Direct Routing: When two computers on the same network need to communicate, the packets do not need a router.
- Indirect Routing: If two computers that are not on the same physical network need to communicate, they must send the Internet Protocol (IP) packet to a router for delivery. Whenever a router is involved in communication, the activity is considered indirect routing. **[PAR 99]**

2-5 TCP/IP Protocol Architecture

TCP/IP consist of four layers, built on top of a physical layer (hardware layer), have to date proven sufficient for all practical purposes. Figure 2.2 shows the TCP/IP model layers from top down.

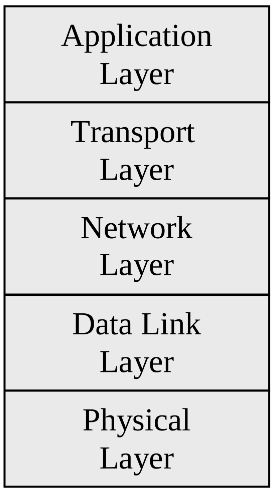


Figure 2.2 TCP/IP Layers

2-5-1 Application Layer Protocols

This is where the user interacts with the network application. Data is received as commands from the user and as data from the network application on the other end of the connection. TCP/IP applications communicate in client/server pairs at this layer. **[PET 99]**

The following protocols are used at application layer:

- **Simple Mail Transfer Protocol (SMTP)**

The simple mail transfer protocol provides a basic electronic mail facility. It provides a mechanism for transferring messages among separate hosts. Features of SMTP include mailing lists, return receipts, and forwarding. The SMTP does not specify the way in which messages are to be created; some local editing or native electronic mail facility is required. Once a message received, SMTP accepts the message and makes use of TCP to send it to a SMTP module on the other host. The target SMTP module will make use of a local electronic mail package to store the incoming message in a user's mailbox. **[WIL 98]**

- **File Transfer Protocol (FTP)**

File transfer protocol is a program that allows files to be copied from one host to another. Less automatic and transparent to the end user than the resource sharing protocols; FTP offers the highest degree of interoperability between hosts. Although FTP requires that files be copied from one host to another before they can be used, it does allow any host running TCP/IP to access a FTP server and exchange data. The two hosts can exchange files regardless of their operating systems, file structure, or even character sets in use. **[PET 99]**

FTP offers an efficient and quick way to transfer files because it does not have the overhead of encoding and decoding data, such as sending files as e-mail attachments. **[PAR 99]**

- **TELNET**

The client TELNET protocol/process is accessed through the local operating system either by a user application or, more usually, by a user at a terminal. It provides services to enable a user to log on to the operating system of a remote machine. All commands (control characters) and data entered at the user terminal-or submitted by the user application program are passed by the local operating system to the client TELNET process which then passes them, using the reliable stream service provided by TCP, to the correspondent server TELNET. The latter issues the commands on behalf of the user, through the local operating system, to the interactive process, as illustrated in the figure 2.3. **[FRE 96]**

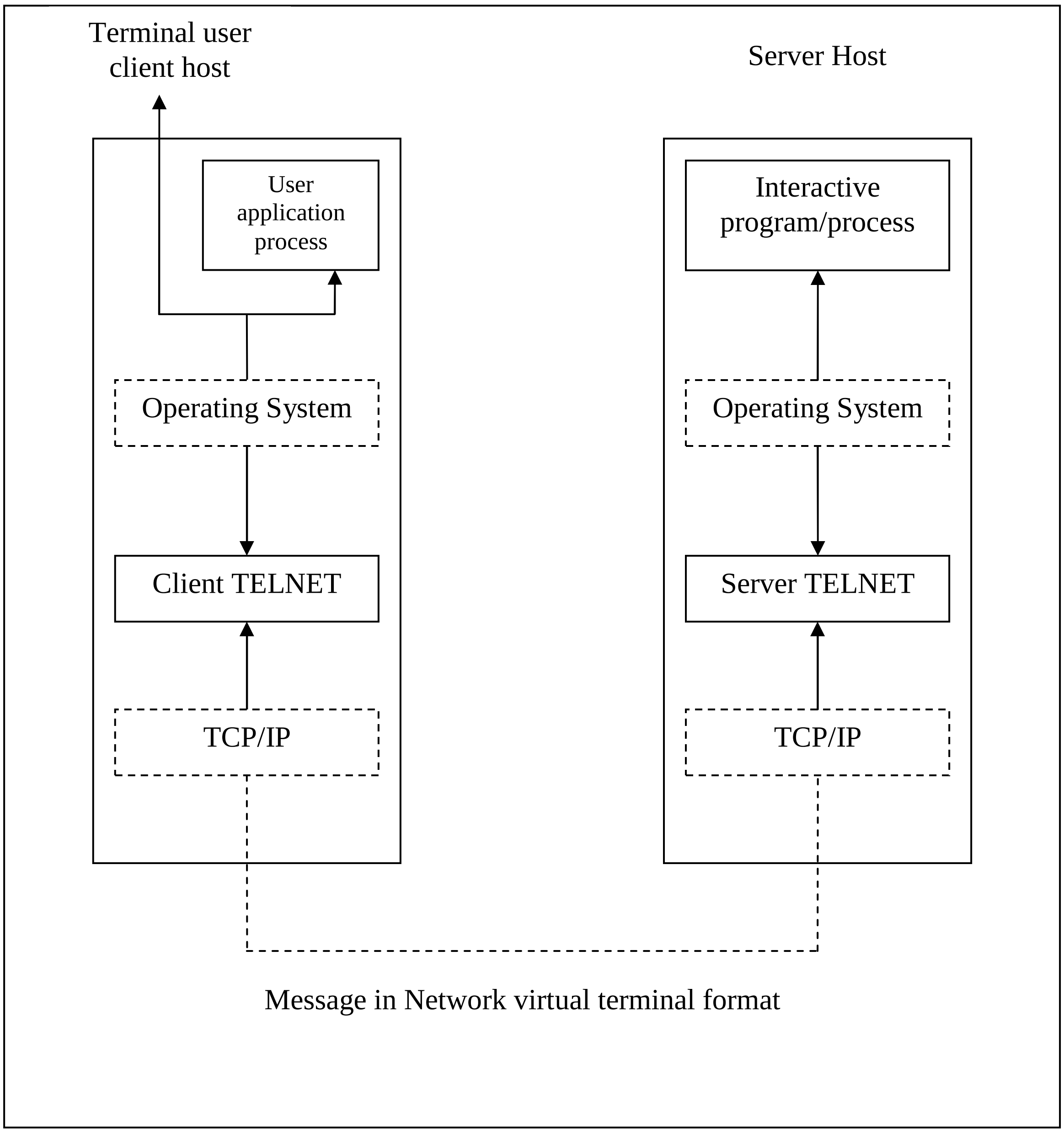


Figure 2.3 TELNET client/server interaction scheme

- **Simple Network Management Protocol (SNMP)**

Simple network management protocol maintains image of the network in database called management information base (MIB). The consistency between the image and the network is maintained by exchanging messages. When a management application modifies attributes of the object in the database, messages are sent to the physical device represented by the object to modify its corresponding attributes. Conversely, the database can learn the attributes of the physical device by polling the devices periodically or by having the devices send relevant information about selected events. **[JEA 91]**

2-5-2 Transport Layer Protocols

This layer manages the flow of data between two internetwork hosts. TCP/IP relies on two transport protocols, TCP (Transmission Control Protocol) for reliable data flow, and UDP (User Datagram Protocol) is a much simpler protocol that offers no reliability guarantees. **[PET 99]**

The header format for TCP, as shown in figure (2.4), which is a minimum of 20 octets, or 160 bits as follow: **[WIL 98]**

- The source port and destination port fields identify the applications at the source and destination systems that are using this connection.
- The sequence, acknowledgment number, and window fields provides flow control and error control.
- The checksum is a 16-bit frame check sequence used to detect errors in the TCP segment.

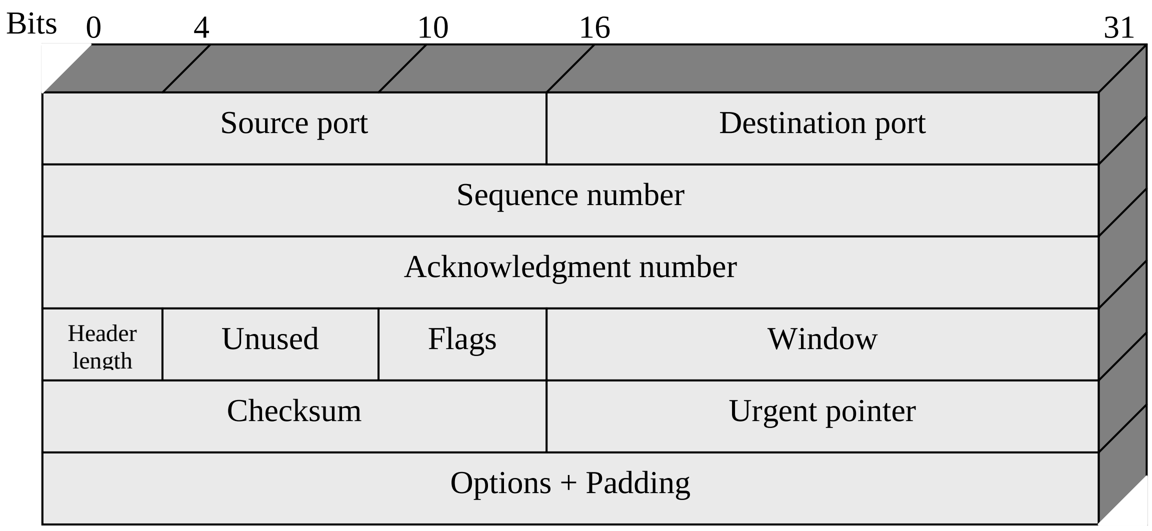
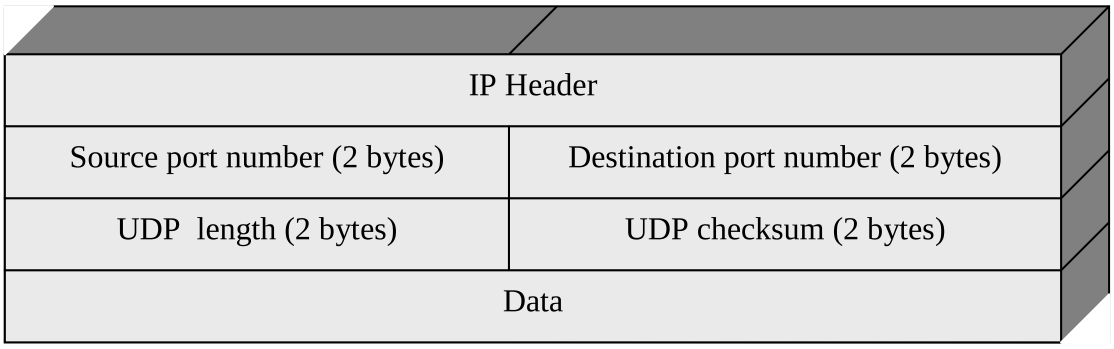


Figure 2.4 TCP Header

The UDP provides a connectionless service for application-level procedure; it does not guarantee delivery, preservation of sequence, or protection against duplication. UDP enables a procedure to send messages to other procedures with a minimum of protocol mechanism. **[WIL 98]**

The UDP header is always eight bytes long, and consists of four two -byte fields, its fields as following: **[PET 99]**

- The first field is the source port number.
- The second field is the destination port number.
- The third field hosts the length of the UDP datagram.
- The fourth field contains a UDP checksum, as illustrated in figure 2.5.



IP Header	
Source port number (2 bytes)	Destination port number (2 bytes)
UDP length (2 bytes)	UDP checksum (2 bytes)
Data	

Figure 2.5 UDP Header

2-5-3 Network Layer Protocols

Data is moved around the internetwork at this layer. The Internet Protocol (IP) operates at this layer to route packets across networks independent of network medium. **[PET 99]**

The IP provides a number of core functions and associated procedures to carry out the various harmonizing functions that are necessary when interworking across dissimilar networks. These include the following: **[FRE 96]**

- **Fragmentation and reassembly**: This concerned the transfer of user messages across networks/subnets which support smaller packet sizes than the user data.
- **Routing**: To perform the routing function, the IP in each source host must know the location of the internet gateway or local router that is attached to the same network or subnet. Also, the IP in each gateway must know the route to be followed to reach other networks or subnets.

- **Error reporting**: When routing or reassembling datagram within a host or gateway, the IP may discard some datagram. This function is concerned with reporting such occurrences back to the IP in the source host and with a number of other reporting functions.

2-5-4 Data Link Layer Protocols

At this layer, also known as the *network interface layer,* data is transmitted across a single network. Data from the network layer, which has been routed appropriately, has arrived at (or has never left) the local network and is transmitted to its destination. **[PET 99]**

This layer is concerned with the exchange of data between an end system and the network to which it is attached. The sending computer must provide the network with the address of the destination computer, so that the network may route the data to the appropriate destination. The sending computer may wish to invoke certain services, such as priority, that might be provided by the network. **[WIL 98]**

2-5-5 Physical Layer Protocols

The physical layer covers the physical interface between a data transmission device (e.g. work station, computer) and a transmission medium or network. This layer is concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, and related matters. **[WIL 98]**

2-6 Client/Server Computing

Most operating systems and networks have basic task-to-task communication facilities. However, the communications would not necessarily occur between similar computers. A PC might need to initiate a conversation with a midrange, or a mainframe might need to communicate with a PC. This was the interesting twist--how to implement a distributed processing environment that took advantage of computing power wherever it was in the network, without requiring all of the computers to use the same operating system or even the same primary networking services. What formed as a possible solution to this puzzle was the concept of client/server computing.

In the client/server scenario, the local computer (PC or a user's session on a larger computer) acts as the processing client. Associated with the client is software that provides a universal appearance to the user (be it a graphical, icon-oriented display, or a menu-oriented display). From that display, you can select the applications you want to use.

When a user selects an application, the client initiates a conversation with the server for that application. This might involve communications across LANs and WANs or simply a call to a local program. Regardless of where the server resides, the client acts as the front end for the server and handles the user interface. Thus, the user is not aware of where the application actually resides. **[CIS 99]**

2-7 File Servers

Computers that act as shared repositories for files are called file servers. File servers provide controlled access to which files and other system resources. The primary purpose of file server software is to synchronize access to shared resources. This means the server software, in cooperation with applications programs, makes sure that users have simultaneous file access where appropriate, while preventing simultaneous access where it is inappropriate.

File server can also provide various levels of security and access control, allowing a system manager to designate who has access to what resources. In this area, there are vast differences in capabilities among various file server systems.

The efficiency and sophistication of a file server's data management and retrieval vary widely from one network operating system to another. High – speed disk access techniques, use of disk caching, and use of proprietary disk file structures are among the methods used to increase data retrieval speed. **[SUR 95]**

2-8 Socket

In a client server model, two application programs, one running on the local system (a client for example) and the other running on the local system (a server for example). To standardize network programming, application-programming interfaces (APIs) have been developed. An API is a set of declarations, definitions, and procedures followed by programmers to write client server programs. Among the more common APIs are the Socket Interface, the Transport Layer Interface (TLI), the Stream Interface, the Thread Interface, and the Remote Procedure Call (RPC). **[BEH 00]**

The socket interface to TCP/IP dates from the early BSD (Berkeley Software Distribution) UNIX systems that first implemented TCP/IP about

1980. It is the primary interface between application programs and the transport layer. The transport layer is usually in the kernel of operating systems whereas higher-level protocols are implemented by programs so the socket interface is usually a set of system calls (although on some systems like Sun Solaris or Windows Winsock it is a library with slightly different transport layer system calls below). **[CS2 01]**

The socket primitives used in Berkeley UNIX for TCP are widely used for Internet programming. They are listed in figure 2.6 The first four primitives in the list are executed in that order by servers. The SOCKET primitive creates a new end point and allocates table space for it within the transport entity. The parameters of the call specify the addressing format to be used, the type of service desired, and the protocol. A successful SOCKET call returns an ordinary file descriptor for use in succeeding calls, the same way an OPEN call does. **[AND 03]**

Primitive	**Meaning**
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 2.6 The socket primitives for TCP

Newly-created sockets do not have network addresses. These are assigned using the BIND primitive. Once a server has bound an address to a socket, remote clients can connect to it. The reason for not having the socket call create an address directly is that some processes care about their address (e.g., they have been using the same address for years and everyone knows this address), whereas others do not care. Next comes the LISTEN call, which allocates-space to queue incoming calls for the case that several clients try to connect at the same time. To block waiting for an incoming connection, the server executes an ACCEPT primitive. When a Transport-DPU (TDPU) asking for a connection arrives, the transport entity creates a new socket with the same properties as the original one and returns a file descriptor for it. The server can then fork off a process or thread to handle the connection on the new socket and go back to waiting for the next connection on the original socket. ACCEPT returns a normal file descriptor, which can be used for reading and writing in the standard way, the same as for files. **[AND 03]**

At client side, a socket must first be created using the SOCKET primitive, but BIND is not required since the address used does not matter to the server. The CONNECT primitive blocks the caller and actively starts the connection process. When it completes (i.e. when the appropriate TPDU is received from the server), the client process is unblocked and the connection is established. Both sides can now use SEND and RECV to transmit and receive data over the full-duplex connection. The standard UNIX READ and WRITE system calls can also be used if none of the special options of SEND and RECV are required. Connection release with sockets is symmetric. When both sides have executed a CLOSE primitive, the connection is released. **[AND 03]**

2-9 LAN Architecture

LAN is typically connects workstations, personal computers, printers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communications between users via electronic mail and other applications. **[BAS 02]**

LANs can be extended by connecting to other similar or dissimilar LANs, to remote users, or to mainframe computers. This process is generally referred to as LAN connectivity. LANs of a particular company can be connected to the LANs of trading partners such as vendors and customers. **[JAM 00]**

2-9-1 LAN Topologies

Whether the purpose of the LAN is to interconnect PCs, minicomputers, or both is almost irrelevant-the first issue is often choosing the topology of the LAN. This choice dictates the cable, cabling methodology and the networking software that can operate on the LAN. The three basic topologies are the ring, star, and bus shown in Figure 2.7.

- **Ring:** As its name suggests, a ring LAN joins a set of attachment units together via a series of point-to-point connections between each unit. Each attachment unit, in turn, interfaces to one or more computers or computing devices. Information flows from attachment unit to attachment unit in a single direction, thus forming a ring network.

Because each PC in a ring network acts as a repeater, performance degrades with each additional PC. Consequently, this is typically appropriate only in small networks.

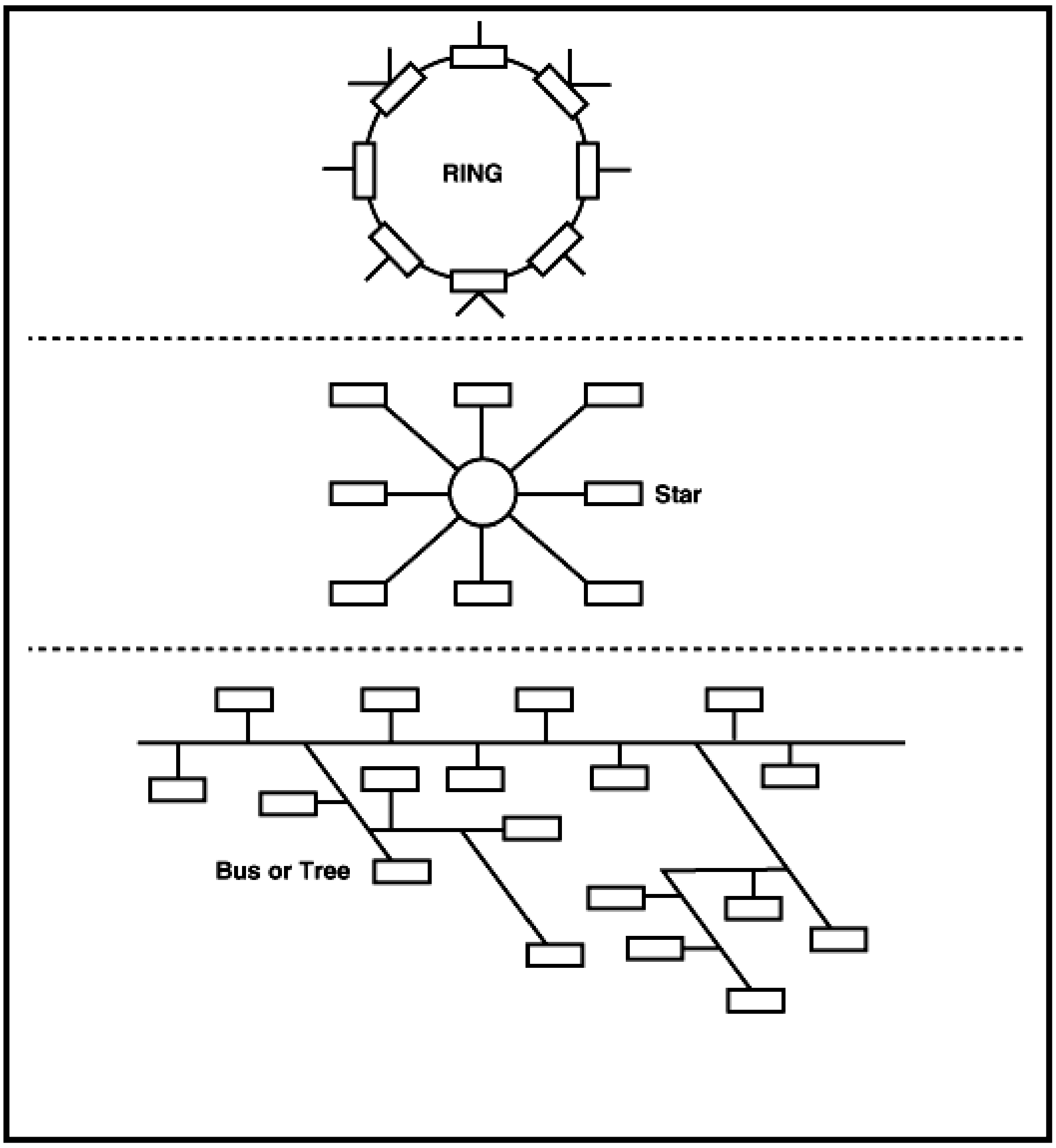


Figure 2.7 LAN Topologies

Star: In a star LAN, each computer or computer-related device is connected on a point-to-point link to a central device called a hub. The hub acts as the LAN traffic manager, setting up communication paths between two devices seeking to exchange information. This configuration makes it very easy to isolate problem nodes, and is one of the most common LAN models. It is easy to install and wire. There is no disruption to the network when connecting or removing devices.

Bus: The simplest form of bus LAN is a set of computers or devices connected to a common, linear connection. Under the bus topology, information is transmitted over the distance of the network, so each computer can pick up its intended information. Links from the main bus line might break off into additional linear links with multiple attachments; this type of bus structure is also referred to as a tree because multiple branches reach out from the main trunk. It is easy to connect a computer or peripheral to a linear bus, requires less cable length than a star topology. The entire network shuts down if there is a break in the main cable. The terminators are required at both ends of the backbone cable. **[JOH 00]**

2-9-2 Wireless LAN

A wireless LAN is a cellular computer network that transmits and receives data with radio signals instead of wires. Wireless LANs are used increasingly in both home and office environments, and public areas such as airports, coffee shops and universities. Innovative ways to utilize WLAN technology are helping people to work and communicate more efficiently. Increased mobility and the absence of cabling and other fixed infrastructure have proven to be beneficial for many users.

Wireless users can use the same applications they use on a wired network. Wireless adapter cards used on laptop and desktop systems support the same protocols as Ethernet adapter cards. **[DAV 01]**

2-10 Cryptography

The discipline relating to the use and development of techniques to encrypt and decrypt messages is called cryptography. An attempt to break a specific cryptography technique is called cryptanalysis. Usually it is assumed that individual only has a copy of an encrypted message when trying to break a specific technique in order to read the message. The process is often made easier if the cryptanalyst can obtain the encrypted version of some known message. The field which covers both cryptography and cryptanalysis is known as cryptology.

The process of encryption entails taking a message (often referred to as plaintext or clear text) and changing it to hide the original meaning from everybody but the intended recipient(s). Decryption is the process that takes the encrypted message (now referred to as ciphertext) and restores it the original message.

This process of changing plaintext to ciphertext and back again requires that two pair of transformations takes place. These transformations use mathematical functions which incorporate an additional piece of data known as the key, to perform the required transformations. The key is kept secret so that only the intended recipient(s) can decrypt the message. These transformations can be represented as follows: **[ERI 00]**

$$\text{Ciphertext} = \text{Encrypt}_{[\text{key}]}\ (\text{Plaintext})$$

$$\text{Plaintext} = \text{Decrypt}_{[\text{key}]}\ (\text{Ciphertext})$$

2-10-1 public key cryptosystem

Public key algorithms rely on one key for encryption and different but related key for decryption. These algorithms have the following important characteristic:

1. It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms such as RSA, also exhibit the following characteristic:

2. Either of the two related keys can be used for encryption, with the other used for decryption.

The essential steps for public key encryption process are following:

1. Each end system in a network generates a pair of keys to be used for encryption and decryption of messages that it will receive.
2. Each system publishes its encryption key by placing it in a public register or file. This is the public key. The companion key kept private.
3. If A wishes to send a message to B, it encrypts the message using B's public key.
4. When B receives the message, B decrypts it using B's private. No other recipient can decrypt the message because only B knows B's private key. **[WIL 99]**

2-10-2 RSA public key encryption

The RSA cryptosystem, named after its inventors R. Rivest, A. Shamir and L. Adleman, was the first public key cryptosystem and is still the most important. Its security is closed related to the difficulty of finding the factorization of a composite positive integer that is the product of two large primes. **[JOH 01]**

- **Key Generation Algorithm**

Each entity creates an RSA public key and corresponding private key as shown in table 2-1.

Each entity A will do the following:

1. Generate two large random (and distinct) primes p & q, each roughly the same size.
2. Compute $n = p.q$ and $\emptyset\ (n) = (p-1)\ (q-1)$.
3. Select a random integer e, $1< e < \emptyset\ (n)$, such that $gcd\ (e, \emptyset\ (n)) = 1$.
4. compute the unique integer d, $1< d < \emptyset\ (n)$, such that $e\ d \equiv 1\ (mod\ \emptyset\ (n))$.
5. A's public key is (n, e); A's private key is d.

- **RSA Public key Encryption and Decryption**

B encrypts a message m from A, which A decrypts.

1. Encryption – B will do the following:
   a. Obtain A's authentic public key (n, e).
   b. Represent the message as an integer m in the interval $[0, n-1]$.
   c. Compute $c = m^e\ mod\ n$.
2. Decryption – to recover plaintext m from c, A will do the following:
   a. Use the private key d to recover $m = c^d\ mod\ n$. **[AME 97]**

p	q	Ø (n)	n	e	d
7	47	276	329	127	163
11	37	360	407	317	293
13	29	336	377	19	283
19	47	828	893	217	145
19	23	396	437	49	97
31	37	1080	1147	403	67
31	29	840	899	457	193
37	41	1440	1517	677	653
41	43	1680	1763	1061	1661
43	37	1512	1591	737	1313

Table 2-1 RSA key

2-11 Data Compression

Those who use compression software are familiar with terms such as zip, implode, stuffit, diet, and squeeze. These are names of programs or methods for compressing data, names chosen to imply compression. However, such names do not reflect the true nature of data compression. Compressing data is not done by staffing or squeezing it, but by removing, any redundancy cannot be compression. Data with redundancy can be compressed. Data without any redundancy cannot be compressed.

The first type of data is text. Text is an important example of computer data. Many computer applications, such as word processing and software compilation, are nonnumeric; they deal with data whose elementary components are characters of text. The computer can store and process only binary information (zeros and ones), so each character of text must be assigned a binary code. Present day computers use the ASCII code, although more and more computers use the new Unicode. ASCII is a fixed size code where each character is assigned an 8 bit code (the code itself occupies seven of the eight bits, and the eighth is parity, designed to increase the reliability of the code). A fixed size code is a natural choice because it makes it easy for software applications to handle characters of text. On the other hand, a fixed size code is inherently redundant.

In a file of random text, each character is occurred approximately the same number of times. However, files used in practice are rarely random. They contain meaningful text, and the typical English text certain letters, such as "E", "T", and "A" are common, whereas other letters, such as "Z" and "Q" are rare. This explains why the ACII is redundant and points the way to eliminated the redundancy. ASCII is redundant because it assigned to each character, common of rare the same number (eight) of bits. Removing redundancy can be done by assigning variable size code to the characters with short codes assigned to the common characters and long codes assigned to the rare once. This is precisely who Huffman coding works.

The second type of common computer data is digital images. A digital image is rectangular array of colored dots, called pixels. Each pixel is represented in the computer by its color code. In order to simplify the software applications that handle images, the pixels are of the same size. The size of a pixel depends on the number of colors in the images, and this number is normally a power of two. If they are 2^k colors in (n) images, then each pixel is a (k bit) number.

There are two types of redundancy in a digital image. The first type is similar to redundancy in text. In any particular images certain color may dominate, while others may be infrequent. This redundancy can be removed by assigning variable size codes to the pixels, as is done with text. The other type of the redundancy is much more important and is the result of the pixel correlation.

The compressor or encoder is a program that compresses the raw data in the input file and increases an output file with compressed (low redundancy) data. The decomposer or decoder converts in the opposite direction. The term encoding is very general and has wide meaning. The term codec is sometimes used to describe both the encoder and decoder. **[DAV 02]**

2-11-1 Dictionary Methods

Statistical compression methods use a statistical model of the data, so the quality of compression they achieve depends on how the good that model is. Dictionary – based compression methods don’t use a statistical model, nor do they use a variable – size codes. Instead they select strings of symbols and encode each string as a token using a dictionary. The dictionary holds strings of symbols and it may be static or dynamic. The former is permanent, sometimes allowing the addition of strings but no deletions, whereas the latter holds strings previously found in the input file, allowing for additions and deletions of strings as new input is read. **[DAV 02]**

2-11-2 LZW

The original Lempel Ziv approach to data compression was first published in 1977. Terry Welch's refinements to the algorithm were published in 1984. the algorithm is surprisingly simple. In a nutshell, LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters. The code that the LZW algorithm outputs can be of any arbitrary length, but it must have more bits in it than a single character. **[MAR 89]**

The main feature of LZW method is eliminating the second field of a token. An LZW token consists of a pointer to the dictionary. The data structure for the dictionary is a tree. The LZW method starts by initializing the dictionary to all the symbols in the alphabet. In the common case of 8 bit symbols, the first 256 entries of the dictionary (entries 0 through 255) are occupied before any data is input. Because the dictionary is initialized, the next input symbol will always be found in the dictionary. This is why an LZW token can consist of just a pointer and does not have to contain a symbol code as in old version. The principle of LZW is that the encoder inputs symbols one by one and accumulates them in a string I. As long as I is found in the dictionary, the process continues. At a certain point, adding the next symbol x causes the search to fail; string I is in the dictionary but string Ix (symbol x append to I) is not. At this point the encoder

1. Outputs the dictionary pointer that points to string I.

2. Saves string Ix (which is now called a phrase) in the next available dictionary entry.
3. Initialize string I to symbol x.

Figure 2.8 is a pseudo code listing of the algorithm. The λ denotes the empty string and <<a, b>> denotes the concatenation of strings a and b.

The line "append <<di, ch>> to the dictionary" is of special interest. It is clear that in practice, the dictionary may fill up. This line should therefore include a test for a full dictionary and certain actions for the case where it is full.

```
For i: = 0 to 255 do
    append i as a 1-symbol string to the dictionary;
append λ to the dictionary;
di : = dictionary index of λ;
repeat
   read (ch);
   if <<di, ch>> is in the dictionary then
        di := dictionary index of <<di, ch>>;
   else
        output (di);
        append <<di, ch>> to the dictionary;
        di : = dictionary index of ch;
   endif
until end of input;
```

Figure 2-7 The LZW Algorithm

Since the first 256 entries of the dictionary are occupied right from the start, pointers to the dictionary have to be longer than 8 bits. A simple implementation would typically use 16 bit pointers, which allow for a 64k-entry dictionary (where 64 k = 2^{16} = 65,536). Such a dictionary will, of course, fill up very quickly in all but the smallest compression jobs. Another interesting fact about LZW is that strings in the dictionary become only one character longer at a time. It therefore takes a long time to get long strings in the dictionary and thus a chance to achieve really good compression.

3-1 Introduction

Assuming everyone involved in the technology has done his or her job well, the only thing users want to know about are the applications they can use with their networked systems. The most important TCP/IP applications now in use are probably the Word Wide Web and electronic mail. However, there are many other applications available in the TCP/IP protocol suite. Some of them were designed for use by network managers and administrator, and others were designed to be used by end users. Most are described in RFCs (Request for Comment), and the ones deemed useful have been implemented on different platforms and improved over time.

Applications are usually implemented in pairs: the client side and the server side. To be useful, an application must be widely implemented both by network managers as a service on network servers and by users as a client program running on networked workstations.

One of the most basic network applications is the ability to manage files on remote systems. The ability to copy, delete, and move such files continues to be important despite the growing popularity of other applications. The proposed protocol uses TCP for reliability. The proposed protocol moves data between any two hosts independent of operating system file commands and file type or character representation. Data file on an IBM mainframe, using the EBCDIC character set, can be transferred to a Windows PC, Macintosh, or Unix workstation as ASCII files, without any special treatment or need for conversion. **[JEA 91]**

3-2 The Proposed System

The proposed system consists of two subsystems as illustrates in figure 3.1, the client and server systems. It is implemented using java programming language, because java runtime implementations currently available link very tightly with platform-native communications libraries that do most of the difficult, speed-and-memory-intensive work of communication. The java APIs provide a convenient consistent wrapper for these various platforms so that it works on any platform that supports the required java APIs. Java provides two types of sockets TCP and UDP and the proposed system uses TCP socket API.

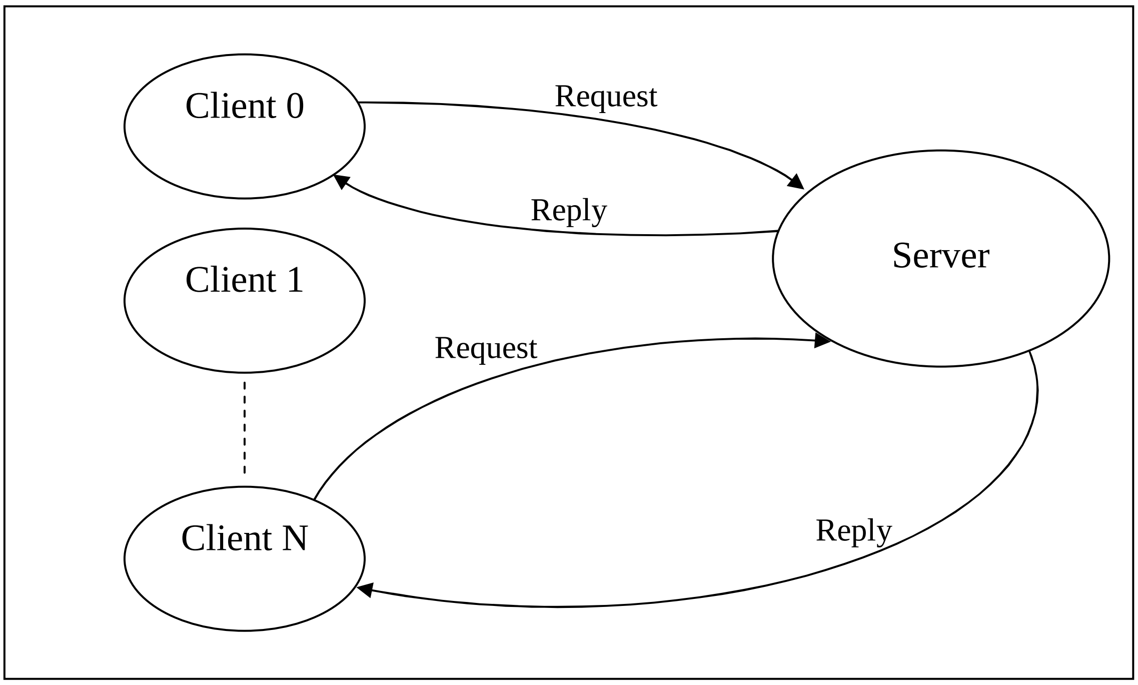


Figure 3.1 client server system

3-2-1 Client System

It is running on the local machine requesting service from a server. A client program is finite, which means it is started by the user and terminates when the service is completed. A client opens the communication channel using the IP address of the remote host and well-known port address of the specific server program running on that machine. After a channel of communication is opened, the client sends its request and receives a response. Although the request-response part may be repeated several times, the whole process is finite and eventually comes to the end. At this moment, the client closes the communication channel with an active close. The client performed a connection to the server used socket API at specified port number, and server name as shown in algorithem 3-1.

```
Socket=new (ServerName,portNo)
DataOutputStream =new (Socket.getOutputStream())
DataInputStream =new (Socket.getInputStream())
```

Algorithm 3-1 client connection

3-2-2 Server System

A server is a program running on the remote machine providing services to the clients. When it started, it opens the door for incoming requests from clients, but it never initiates a service until it is requested to do so. A server program is an infinite program. When it started, it runs infinitely unless a problem arises. It waits for incoming requests from clients. When requests arrived, it respond to the requests concurrently. Server process uses socket API for accept connection with clients at specified port number as illustrated in algorithm 3-2 and it consists of the following operations:-

```
Begin
ServerSocket(portNo)
while(flag)
ServerSocket.accept()
get request from client
if request rejected then send error message
else send back acknowledgment and serve request
end
```

Algorithm 3-2 server process

a. Get Request

The server process receives request from any client, which means a connection to that server, and serving it as a thread (thread is a light weight process), each request consists of user id, file name, and operation as shown in algorithm 3-3.

```
begin
        read user id
        read file name
        read operation
end
```

Algorithm 3-3 get request

b. Process Request

This step checks if that request is acceptable in logic, as shown in algorithm 3-4. Moreover, it checks if the requestor client is authorized to perform this type of request, as it will be illustrated in security section.

```
begin
check does the file  not exist                  (for create request)
        return true
check does the file exist and can read          (for read request)
        return true
check does the file exist and can deleted       (for delete request)
        return true
check does the file exist and can updated       (for update request)
        return true
otherwise return false
end
```

Algorithm 3-4 possible request

c. Send Acknowledgement

After receiving a request from a client, the server checks if the client is authorized to access file, if it is so, then it send back an acknowledgment announcing that it is ready to accept the arguments, otherwise it tells him that his request is reject and terminate.

Another acknowledgment is send to the client when the server finish processing its request and it will terminate the process.

d. Receive Arguments

If the request is accepted then the server receive the client’s arguments for processing.

3-2-3 Server Modules

The server program consists of three modules as shown in figure 3-2:-

- **Manager**

  It represents the head process of the server program, so when a request from any client is received then, it creates a new separate thread to serve that request.

- **Controller**

  For each serving request, there is only one controller, which is responsible for controlling the connection and data communication for the client that sent a request.

- **Service Provider**

  It is consists of many sub modules where each one of them is responsible for providing specific function such as create file, read file, modify file, delete file, generate security key, encrypt data, and decrypt data.

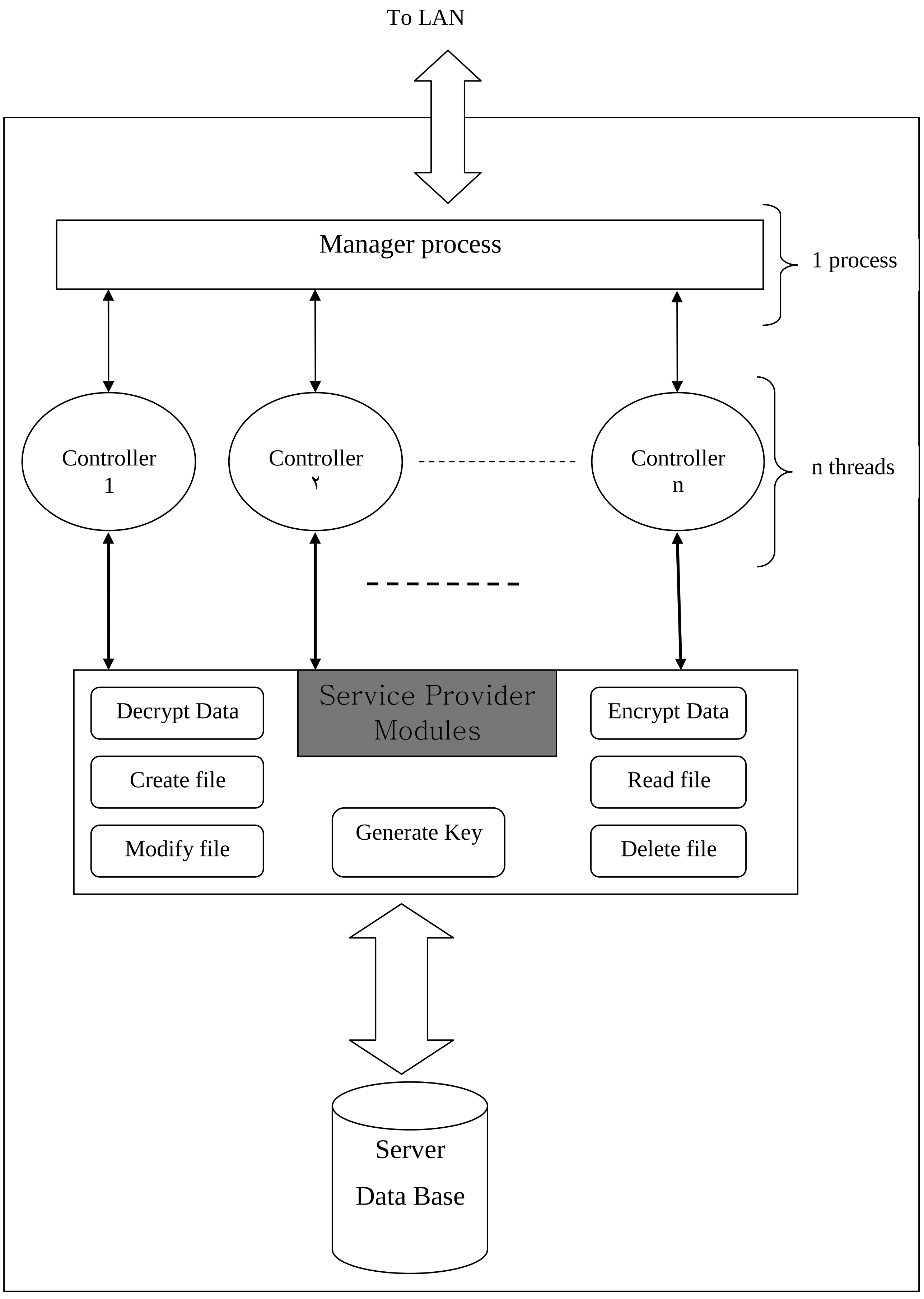


Figure 3-2 Server System

3-3 The Operations of Protocol

The proposed system provides four main file operations to be performed on file, they are as follows:-

1. create file.
2. read file.
3. modify file.
4. delete file.

In the client side, the data will be compressed before transfer, and it will be saved in server database as a compressed version. Also each file could be encrypted in the sending side and decrypted in the receiving side.

3-3-1 Create File

This operation permits the client to create a new file in the server database, the file must have new name, different from other files. Each created file created must have its individual properties, which some of them are set by creator such as *access list*, *account number*, and *available operations*, more properties are illustrated in the virtual file attributes section. In the server side the blocks of data will be saved in new file as shown in algorithm 3-5.

```
begin
do
    read block of data from client
    save bock in a buffer
while more block
write blocks from buffer to the new file
read file attributes from client
write file attributes in database
end
```

Algorithm 3-5 create file

3-3-2 Read File

This operation enables the user to read a file from the server database and save it in his local database with the name that he desired. When the user is authorized to read file, he will get the file. In the server side after data being transferred to client, the virtual file attributes will be modified as shown in algorithm 3-6.

```
begin
open buffer for read
while other block
begin
    read block from file
    add block to buffer
end
while buffer not empty
begin
    read block from buffer
    send block to client
end
assign last file read to client id
assign last date and time of the reading file to the current date and time
end
```

Algorithm 3-6 read file

3-3-3 Modify File

This operation permits the user to modify any file in the server database, (when he authorized to do this operation on this file). In the server side this operation is like *create file* operation, except instead of create new virtual file attributes, it will modify it.

3-3-4 Delete File

This operation enables the user to remove file from the server database, when he is authorized and this operation is permitted to be done on this file.

3-4 The Security of Protocol

The proposed system security includes two steps:-Authentication Access and Data Encryption.

3-4-1 Authentication Access

It helps the owner of a file to specify the number of clients that can access this file and the type of access for each file. There are three types of access:-

1. Read group: this group contains the clients who can read the file.
2. Modify group: this group contains the clients who can modify the file.
3. Delete group: this group contains the clients who can delete the file.

In the proposed system, the owner of the file will be added to the three groups automatically. For the client, to access any file in the server database, he must be authorized (its *id* which is a string previously added to the all groups) to access operation.

3-4-2 Data Encryption

The proposed system uses RSA public key method to encrypt the data before transfer it. It consists of two main operations:- ciphering and deciphering. For transferring data from client to server, the data will be encrypted in the client side and decrypted in the server side and vice versa. Mathematically, encryption and decryption are not different (just in arguments), and can be solved by using *power mod* function as illustrated in algorithm 3-7. In the Public encryption, two keys are used, public key for ciphering and private key for deciphering. The key generation is done in the side who received data.

```
powerMod (base, power, modNumber)
begin
        let base, power, modNumber are integers.
        computes n, where base ^ n > modNumber
        computes i, as
                        while (power > = n)
                                i = i + 1
                                power = power-1
        If  i = 0
                return base ^ power  Mod  modNumber
        Else
          begin
                K1=base^n  Mod modeNumber
                for counter= 1 to i
                  array[counter]=k1
                for counter = 1 to i
                  P = power-n
                If  P > 0  then
                        begin
                            K2 = base ^p  Mod  modNumber
                            add k2 as new cell in array
                        end
                mul = 1 ,  j = 0
                do
                   begin
                        mul = mul * array [ j ]
                        if  mul > = modNumber then
                           begin
                             k = mul  Mod  modNumber
                             AddCell( k )
                              Mul = 1
                            end
                        j++
                   end
                while j< array length
                return mul  Mod  modNumber
          end
end
```

Algorithm 3-7 power mod

3-5 The Compression of Protocol

In the proposed system, the file is compressed (using LZW method) before transferring, and it will be saved in the server as compressed version, when any client want it for manipulation, the client will decompress the file in its side. The compressing operation reduces the file size as shown in table 3-1.

Origin file size in kb	File size after compress in kb
1.03	1.00
2.06	1.59
4.13	2.51
8.26	3.87
16.05	5.93
33.00	8.01
66.01	13.05
132.00	20.00

Tables 3-1 Size of file

The advantage of compressing file is to reduce file size that minimize time transfer for that file. When file compressed, its new size will depends on the nature of that file.

Algorithm 3-8 illustrates the decompress operation, where the decoder starts with the first entries of its dictionary initialized to all the symbols of the alphabet (256 entries). It then reads its input stream (which consists of pointers to the dictionary) and uses each pointer to retrieve uncompress symbols from its dictionary and write them on its output stream. It also builds its dictionary in the same way as the encoder.

```
begin
        for i= 0 to 255
           append i as 1-symbol to dictionary
        read old_code
        output old_code
        ch= old_code
        while more input character
            read new_code
            if new_code is not in dictionary
                ST= get translation of old_code
                ST= ST+ch
            else ST= get translation of new_code
            output ST
            ch= first character in ST
            add old_code+ch to translation table
            old_code = new-code
end
```

Algorithm 3-8 decompress

3-6 The Synchronization of Protocol

To synchronize access to server database, the server includes two shared data structure:-

1-Lock List

It is an array of elements, each element refers to a file name. The request needs one specific file to serve it, when the file is in Lock List, then the request will blocked, until the specified file exit from the Lock List. When the request is create file, then it will insert the file name in the Lock List to prevent any other client to create, read, modify, or delete a file with identical name. When the request is modify file, then the specified file name will insert to the Lock List to prevent any client to request it. In addition, delete file request will insert the file to the Lock List. The request will remove the file (after insert it) from the Lock List when completed. The operations of inserting and removing are performed critically.

2-Read List

It is an array of objects, each object consists of a file name and a List of client’s id, those who are reading the file, as shown in figure 3-3. Each read request will insert the file to the Read List, when complete, it will remove it.

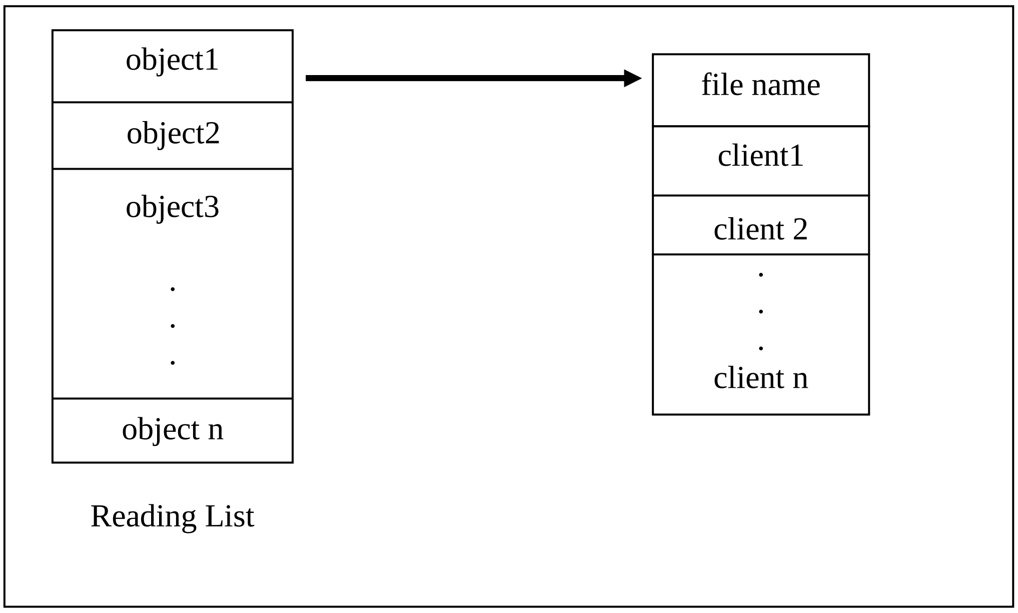


Figure 3-3 Reading List

So, when client1 reading file1, the operation of reading is done by copy all blocks of file1 to buffer1, then client1 read from buffer1, and when client2 wants to modify file1, it will write new data to the buffer2 and after complete, the data will written to file1, during this if client3 wants to read file1 then:

1. If client2 not finish, client3 will read from buffer1, as shown in figure 3-13 a.
2. If client2 and client1 are finish, then client3 will open new buffer3 to read from it, but when client2 finish and client1 not finish, then waiting until client1 finish, then open new buffer, as shown in figure 3-13 b.

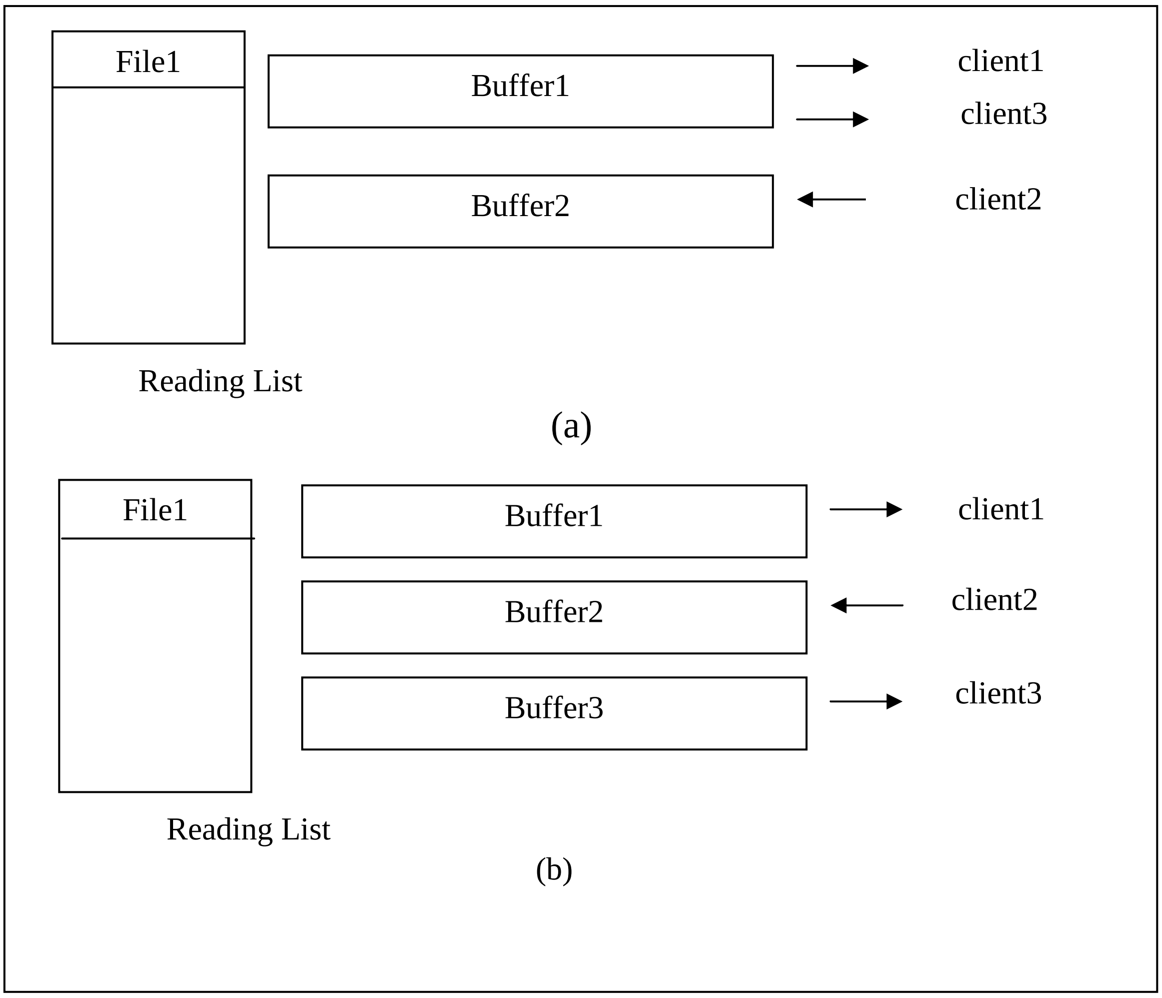


Figure 3-4 client synchronization

3-7 Virtual File Attributes

All files have attributes that describes them. Each must have a name r, and a size telling how much storage it currently occupies. Figure 3.3 shows the proposed system virtual file store attributes. Each attribute has name, type, and value.

Some attributes are created when the file is created, and are forever frozen thereafter. Others can be explicitly changed by user operations. Still other (e.g., time of last modification) are automatically maintained by the file server.

Although most of last the attributes are straightforward, a few of them require some comment. The *Allowed Operations* attribute allows the creator of a file to specify, for example, that some operations are not valid on this file. *Access Control* determines who may access the file, and how.

Attribute	Type	Set at file created	User changeable	Change by server
File Name	String	Yes	Yes	No
Allowed Operation	Boolean	Yes	No	No
Access Control	List	No	Yes	No
Account Number	Integer	Yes	Yes	No
Time and Date of file creation	Time	Yes	No	No
Time and Date of last file modifier	Time	Yes	No	Yes
Time and Date of last file read	Time	Yes	No	Yes
Owner	User ID	Yes	No	No
Last Modifier (user ID)	User ID	Yes	No	Yes
Last Reader	User ID	Yes	No	Yes
Last Attribute Modifier	User ID	Yes	No	Yes
Size	LongInt	Yes	No	Yes

Figure 3.5 File Virtual Attributes

4-1 Proposed TFTP Interface

The proposed system includes two sides of transfer,

1. Client side.
2. Server side.

For each side there are some windows, which are helping the user to get good interaction with the protocol.

4-2 Client Side

After starting up the system, the user must enter his **id**, which specify its limitations for accessing files, then the user can choose the type of data transfer either secure transfer or none secure transfer, as shown in figure 4.1, which is called security frame. It contains a text field for user id and two check boxes for secure transfer or not.

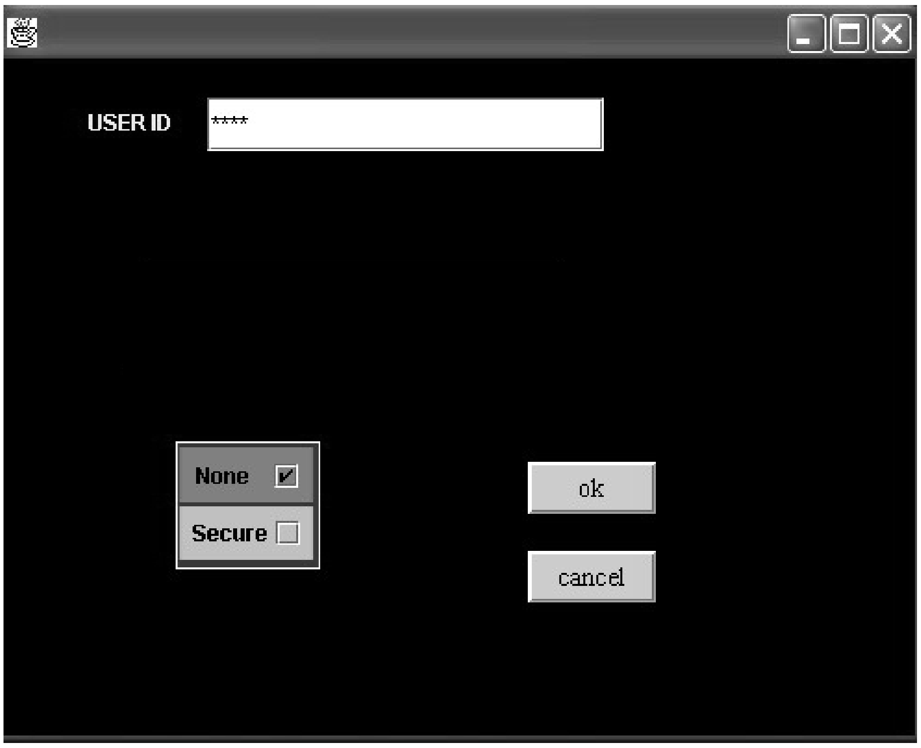


Figure 4.1 The Security Frame

When the system getting the user **id** and the transfer type (secure or not), another frame with the main operations will be appeared, these are:

1. Create file
2. Read file
3. Modify file
4. Delete file

After executing any operation, a message indicate the success of operation or an error message with its error type will appear.

4-2-1 Create New File Operation

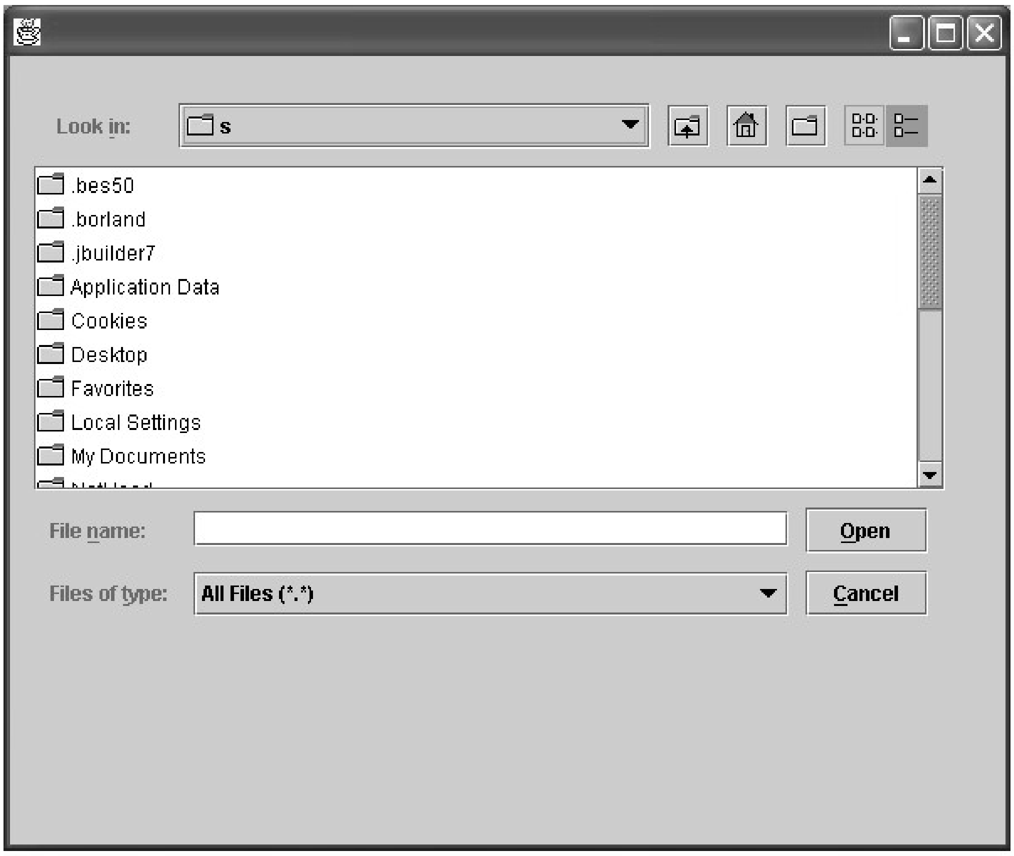
At this operation, the user must fill some information about the new file, as shown in figure 4-3.

- **File Name**

  For this bottom, the user can choose an existing file from the open menu, see figure 4-2.

- **Account Number**

  It is the second field and the user can fill it by writing the account number at a text field, as shown in figure 4-3.



Look in: s

File name:

Files of type: All Files (*.*)

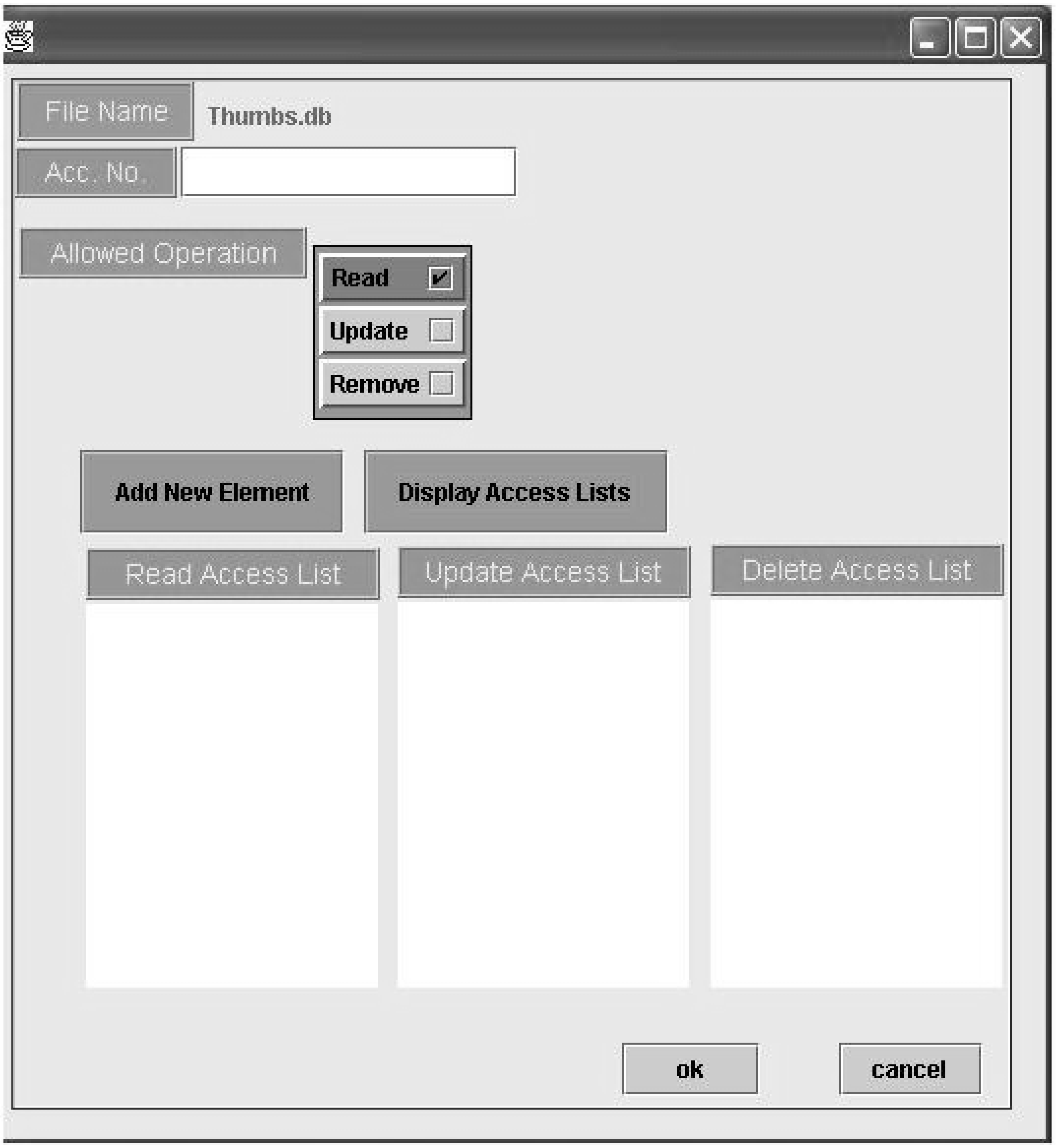
Figure 4.2 Open Menu

- **Allowed Operations**

  Three check boxes for read, update, and remove could be specified to the chosen file name. When the check box is chosen then it will be in highlighted, as shown in figure 4-3.

- **Display Access List**

  There are three lists, each of them represents the set of users who they can perform the specific operation on the selected file, as shown in figure 4-3.



File Name Thumbs.db

Acc. No.

Allowed Operation: Read ✓, Update, Remove

Figure 4.3 Filling Main Frame

- **Add new Element**

  This bottom is clicked when there is new client, additional information is needed to complete this operation such as client id, user properties and the client permitted operation by selecting one or more of the three operation bottoms, as shown in figure 4-4.

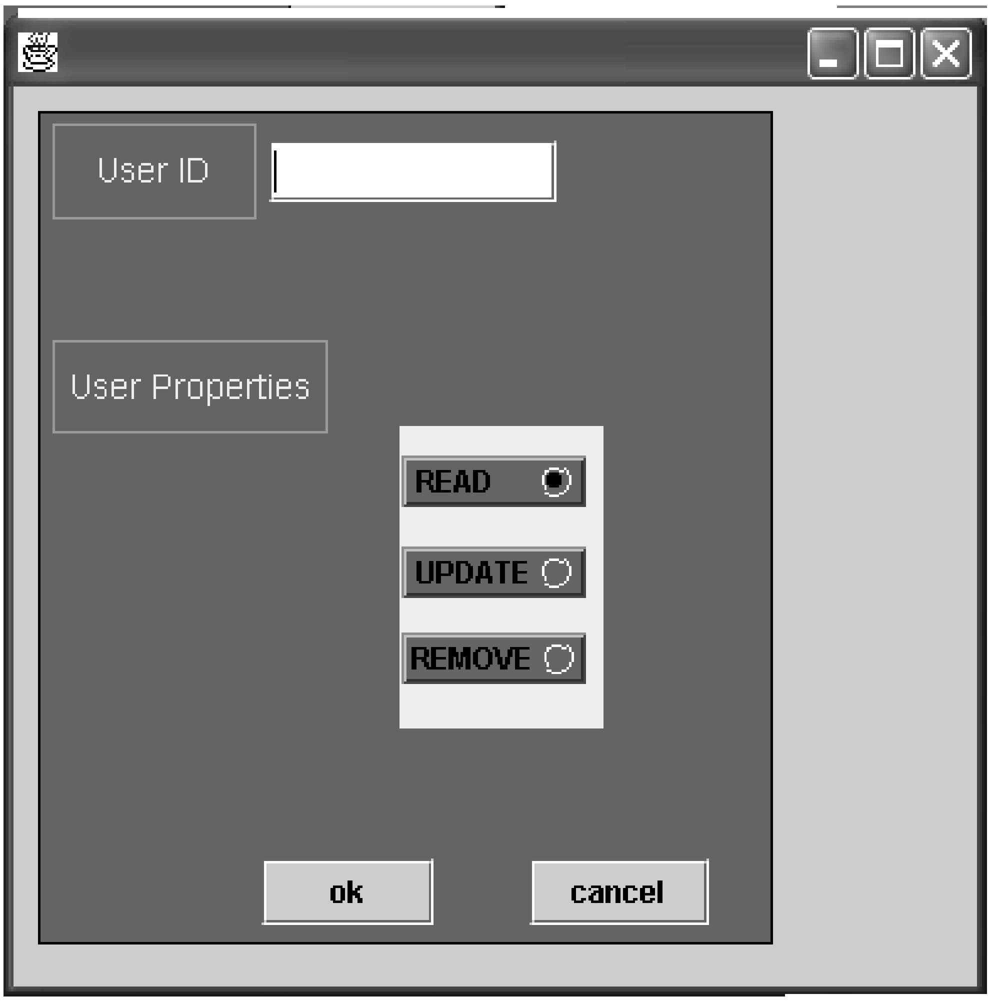
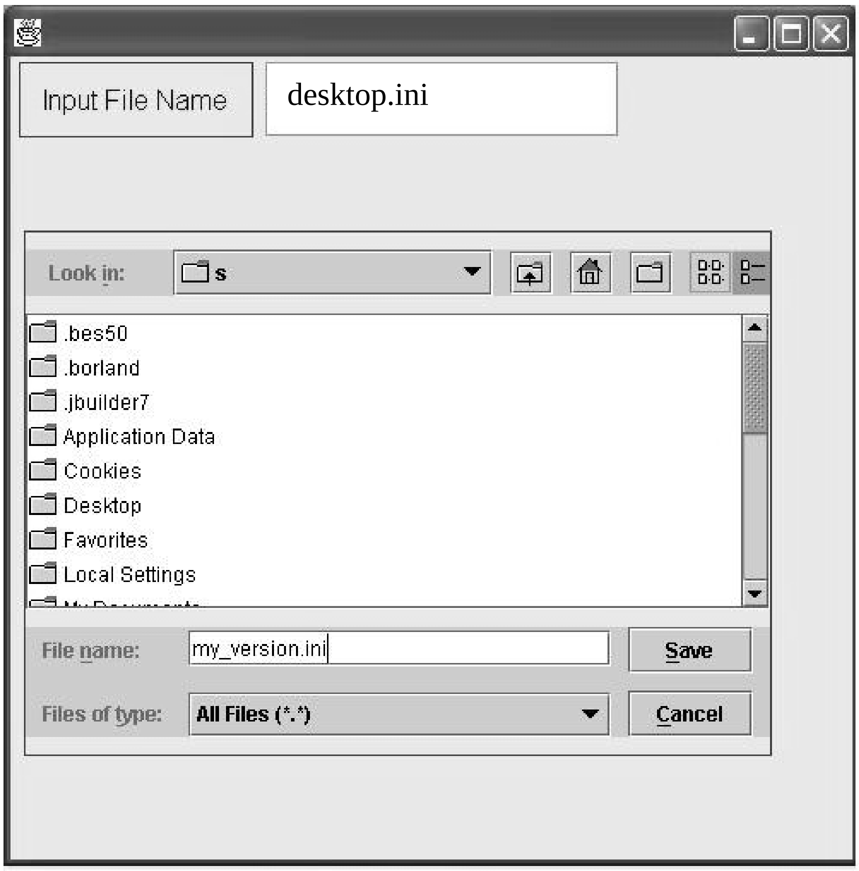


Figure 4.4 client permitted operations

4-2-2 Read File Operation

For reading a file from a server database and saving it in the client side, the user must select the name of the file which is saved in server data base, then choose the file location and write new name for file at save menu, as shown in figure 4-5.


Input File Name desktop.ini

Look in: s

.bes50
.borland
.jbuilder7
Application Data
Cookies
Desktop
Favorites
Local Settings

File name: my_version.ini

Files of type: All Files (*.*)

Figure 4-5 Save Menu

4-2-3 Modify File Operation

To modify the data of a file at server database, the user select a file name, the server must check if this user has the authenticity for updating, if so, the user will be able for updating and then saving the file, otherwise error message will appear, as shown in figure 4-5.

4-2-4 Delete File Operation

To delete any file from server database the user must write the name of the file to be deleted at a text field, which is appeared in the frame, as shown in the figure 4-6.

4-3 Server Side

In the server side, the interface contains only one main frame that includes Lock List, which contains the names of the locked files which cannot be accessed by any client. It also contains information about the current requests as user id, operation name, and file name, as illustrated in figure 4-7.

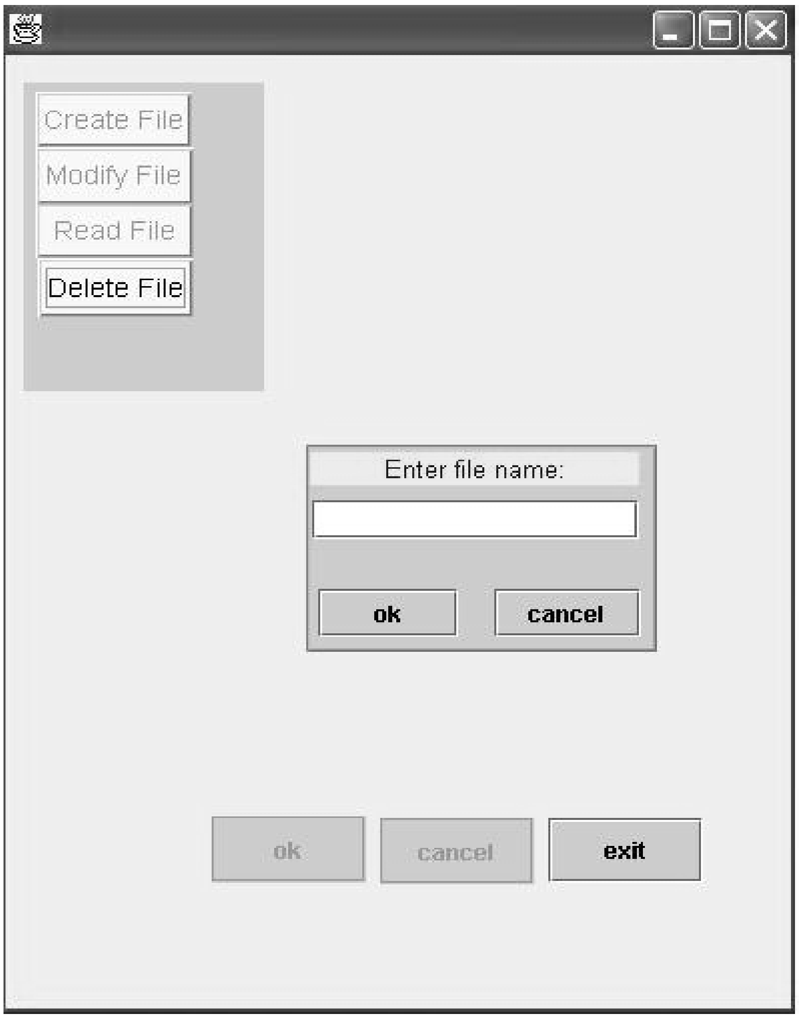


Figure 4-6 Delete Operation Interface

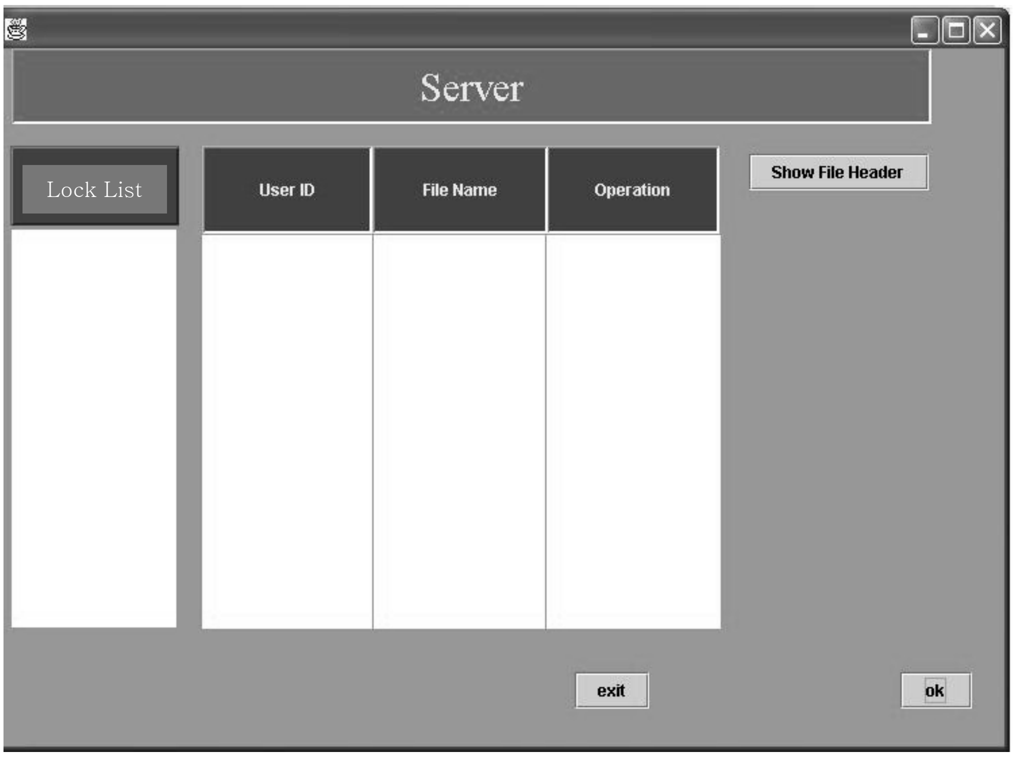
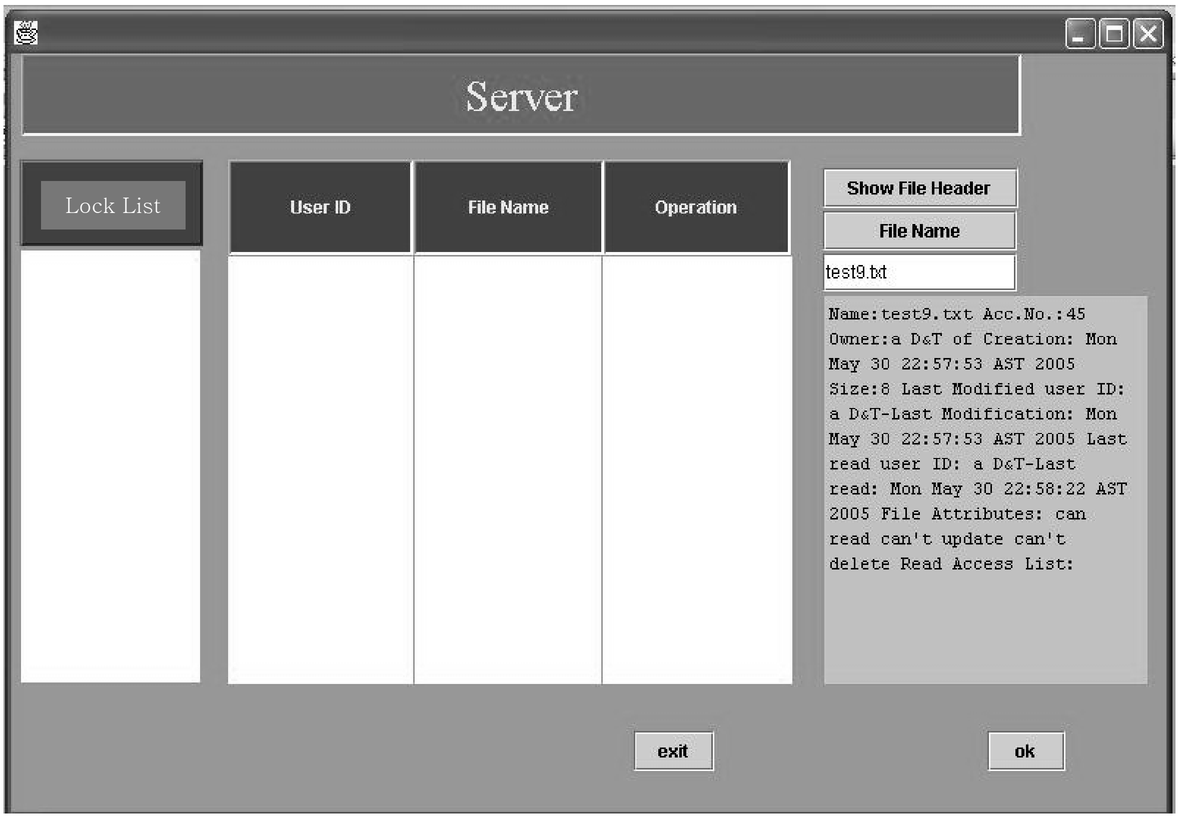


Figure 4-7 Server Frame

The server frame includes a Show File Header bottom that enables the user to show file virtual attributes. When a Show File Header bottom is chosen then one bottom and one text field will appear. The text field used to display the file name, and the bottom used to choose the file name from server database, as shown in figure 4-8.



test9.txt

```
Name:test9.txt Acc.No.:45
Owner:a D&T of Creation: Mon
May 30 22:57:53 AST 2005
Size:8 Last Modified user ID:
a D&T-Last Modification: Mon
May 30 22:57:53 AST 2005 Last
read user ID: a D&T-Last
read: Mon May 30 22:58:22 AST
2005 File Attributes: can
read can't update can't
delete Read Access List:
```

Figure 4-8 Display File Attribute

5-1 Conclusions

1. The proposed protocol has a security for access the data and for data transfer, so it increase security measurement over time.
2. The proposed protocol use one of the new data compression method which is LZW that is efficient since it does not need to pass the string table to the decompress code, also it provides a compression rate so minimizing transferring time.
3. The proposed protocol is a one application of the TCP/IP reference model, it is linked with it by using socket API, and it has the most properties that TCP/IP provides to its application, in addition to the encryption, decryption, and compression capabilities.

5-2 Future Works

1. Develop the protocol to transfer all file types such as image files.
2. Work on distributed database and using more than one server.
3. Build a new protocol on the UDP.

[AME 97]

Menezes, P. Oorschot and S. Vanstone,’’ Hand book of applied cryptography’’, by CRC press, 1997.

[AND 03]

Andrew S. Tanenbaum,’’ Computer Networks’’, Fourth Edition, Prentice Hall, 2003.

[AND 89]

Andrew S. Tanenbaum,’’ Computer Networks’’, Second Edition, Prentice-Hall International, 1989.

[BAS 02]

Bassam S. S.,’’ Design and Implementation of an Online Cryptography System for LANs’’, MSc. Thesis , Saddam University, college of Engineering, 2002.

[BEH 00]

Behrouz A. Forouzan and Sophia Chung Fegan,’’ TCP/IP protocol suit’’, McGraw-Hill, 2000.

[CIS 99]

Cisco Networking Academy Program English-Semester Online curriculum, CD collection Release Version 2.0 copyright Cisco system, 1999.

[CS2 01]

CS2 & N2 NETWORKS NOTES 2000-2001,’’ Network programming with sockets & TCP/IP’’.

[DAV 02]

David Salomon,’’ A Guide to Data Compression Methods’’, Springer –Verlage New York, 2002.

[DAV 01]

David A.,’’ Local Area Networks’’, Third Edition, Prentice Hall, 2001.

[DAV 89]

David H.,’’ Local Area Network Architecture’’, Addison – Wesley, 1989.

[ERI 00]

Eric A. Fisch And Gregory B. White,’’ Secure Computers And Networks’’, Prentice by CRC, 2000.

[ESR 00]

AL-Namemi E. I. S.,’’ Using Hashing and RSA Algorithms for Log-in Authentication’’, MSc. Thesis, Saddam University college of Science, 2000.

[FRE 96]

Fred Halsall,’’ Data Communications, Computer Networks and Open Systems’’, Fourth Edition, Addison-Wesley, 1996.

[JAM 00]

James E. and Philip T.,’’ Local Area Networks’’, Second Edition, Jone Wiley and sons, 2000.

[JEA 91]

Jean Watrand,’’ Communication Networks’’, Addison–Wesley, 1991.

[JOH 01]

Johannes A. Buchmann,’’ Introduction to cryptography’’, Springer, 2001.

[JOH 00]

John E. and Dan W.,’’ Managing Multivendor Networks’’, Macmillan, Computer Publishing USA, 2000.

[MAR 89]

Mark Nelson,’’ LZW Data Compression’’, 1989, http:/www.dogma.net/markn/articles/lzw/lzw.htm

[PAR 99]

Partrick T. and James S., ICII (Internet Certification Institute International),’’ Network Fundamentals Book1’’, 1999.

[PET 99]

Pete L.,’’ TCP/IP Clearly Explained’’, Third edition, printed by Academic Press, 1999.

[SAM 90]

Sam C. and Steve M.,” Mass Storage System Reference Model: Version 4”, Developed by the IEEE Technical Committee on Mass Storage Systems and Technology, ١٩٩٠.

[SUR 95]

Suresh Basandra,’’ Local area network’’, Galgotia Publications pvt. Ltd, 1995.

[WIL 99]

W. Stallings,’’ Cryptography and network security’’, second edition, Prentice hall, 1999.

[WIL 98]

W. Stallings,’’ High-Speed Networks: TCP/IP and ATM Design Principles’’, Prentice-Hall, 1998**.**

[YAS 92]

Yassen T.,’’ Text Compression Computer Models’’, MSc. Thesis, Saddam University college of Science, 1992.

اسم الطالب: سيف محمود خلف حسين العلاك   Saif Mahmood Khlaf

الكلية والقسم : العلوم / الحاسوب

العنوان : بابل – ناحية المدحتية – محلة ٩ – زقاق ١٢ – دار ٧

رقم الهاتف : ٠٣٠,٤٦٣٤٠١

رقم الموبايل : ٠٧٩٠٢١٩٦٥٨٢

البريد الالكتروني : saif463@yahoo.com

saif.shareefy.gmail

عنوان الاطروحة :

تصميم وتنفيذ تطبيق لنقل الملف النصي باستخدام نموذج الـ (TCP/IP)

Design and Implementation of Text File Transfer Protocol Using TCP/IP Reference Model

تاريخ المناقشة : ٣١ / ٥ / ٢٠٠٥