P-1721

DISTRIBUTED FILE SHARING SYSTEM

By

S.JAYAGANESAN
Reg. No 71203621017

of

KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore
(Affiliated to Anna University)

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree

of

MASTER OF COMPUTER APPLICATION

June, 2006

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641023

Department of Computer Applications

Bonafide Certificate

Certified that this project report titled **DISTRIBUTED FILE SHARING SYSTEM** is the bonafide work of Mr. **S.JAYAGANESAN (Reg No.71203621017)** who carried out the research under my supervision. Certified further that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible] 9/6/06 | [illegible]
PROJECT GUIDE | **HEAD OF THE DEPARTMENT**

Submitted for the University Examination held on 30.6.06

[illegible] 30/6 | [illegible] 30/6/06
INTERNAL EXAMINER | **EXTERNAL EXAMINER**

Aadhityaa Infomedia Solutions

New # 77/2, Old # 26/2, Habibullah Road, T.Nagar, Chennai - 600 017.
Voice : +91- 44 - 2822 7223, 2822 7224, 2822 7225
E-mail : aadhityaa@sancharnet.in, info@aadhityaa.com
URL : www.aadhityaa.com

AAD/PR/9856/05-06 29/05/06

To
Head of The Department
Department of Master of Computer Applications,
Kumaraguru College of Technology,
Coimbatore.

Sir,

Sub: **Project Completion**

This is to Certify that **Mr. S.JAYAGANESAN** of **MCA** (Master of Computer Applications) of **KUMARAGURU COLLEGE OF TECHNOLOGY** is successfully completed his project entitled **"DISTRIBUTED FILE SHARING SYSTEM"** during the period of **December 2005 to May 2006** in our company.

For Aadhityaa infomedia Solutions,

[illegible]

Administration-Manager.

ABSTRACT

The project titled **"DISTRIBUTED FILE SHARING SYSTEM"** is codenamed as **DFS**. DFS is a Intranet based Application software that distributes the files among the clients. The project is developed for **AADHITIYA INFOMEDIA SOLUTIONS** situated in Chennai. The project is done with Windows 2000 operating system using Java technologies as front end and ORACLE as backend.

There has been a huge growth in the use of file sharing software over the last year, making file sharing one of the top uses of the Internet. Napster was the first technology that empowered users with the ability to share files among themselves. The demise of Napster led to the development of numerous file sharing technologies, most of which are based on the Gnutella protocol. The problem with all current file sharing technologies is that none of them can guarantee the security of the shared file. This report describes the design, development and evaluation of a file sharing system that proposes a novel solution to the shared file security problem. The system will allow users to share files in a secure manner and comprises of client and server applications.

The client allows users to connect to a server and shares their files among all other system users. The client also gives users the ability to search for files shared by the other system users, and when a file is found it could be transferred from the other user securely, as the file would be encrypted. The server allows valid users to connect, records their shared file list and enables connected users to search this list. The server authenticates connected users by performing a test that only a valid user can respond to in the correct manner.

The report outlines research in the area of networking, including technological backgrounds such as distributed file system architectures, cryptographic techniques and network programming methods. The novel feature of the system is the method used to address the security problem inherent to all current file sharing technologies. The system developed uses cryptographic techniques to implement a framework in which to model system security, including authentication of system users and the files that they share.

ACKNOWLEDGEMENT

I wish to express my sincere thanks to **Dr.Joseph V.Thanikal,** Principal, Kumaraguru College of Technology, Coimbatore, for permitting me to undertake this project.

My deepest acknowledgement to **Dr.M.Gururajan,** HOD, Department, of Computer Applications, Kumaraguru College of Technology, Coimbatore, for his timely help and guidance throughout this project.

I am greatly indebted to my guide **Mrs.P.Parameswari,** Senior Lecturer, Department of Computer Applications, Kumaraguru College of Technology, for her valuable guidance and encouragement at every stage of this project work.

I express my sincere thanks to **Mr.D.Rajan**, Managing Director, **AADHITIYA INFOMEDIA SOLUTION, Chennai** for giving me the opportunity to do the project work in their concern. My deepest acknowledgement to **Mr.S.Prabhu** Project Manager's AADHITIYA INFOMEDIA SOLUTION, for his support and assistance at various levels of my project work.

I wish to thank all my staff members for their timely help and guidance to complete the project successfully.

Finally, my heartfelt thanks to almighty god for helping me throughout the course of this project work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE ORGANISATION

AADHITIYA INFOMEDIA SOLUTIONS is a SOLUTION PROVIDER in the field of Information Technology providing quality solutions in the area of Local Area Networking, Wide Area Networking, software development, implementation, systems integration and data capture solutions, web based and multimedia solutions. Established in 2000.

Landmark InfoTech commenced operations with the setting up of a software training division and software development centre in India with the first centre at Chennai and also offers a total end to solutions in Networking, System Integration & implementation, Automatic data capture systems and Biometric solutions.

Aadhitiya Infomedia Solutions is a part of the 25 years old Landmark Group, which operates large Marketing Management in various parts of the world. Landmark group give important once to the Customers and non-customers.

Landmark Group has been operating large financial companies in various parts of the world and predominantly in the Middle East. Landmark Group has ambitious global expansion plans and is gearing up to be one of the large Insurance groups in the world. The Landmark's IT division has been spinning off into a separate commercial venture with the concept of application oriented Information technology products and services to care the requirements of the IT industry at large.

SOLUTIONS OFFERED

AADHITIYA has in its portfolio the following solutions:

- Wireless Data Capture Solutions
- Biometric Based Solutions
- Security Solutions
- Retail Management Solutions for small, medium and large retail outlets
- Warehouse Management and Logistics
- B2B Collaborative services
- Financial Management Solutions
- Networking solutions both wired and wireless
- Internet, Intranet, Extranet and other Web Based Solutions.
- Hardware and Networking

The Vision

To be the most vociferous proponents of application based information technology by designing, developing, implementing and providing products, solutions and services with business requirements as the basis of technology innovation.

The Mission

Take every domain and provide need based information technology products, solutions and services using experts with hands on real world experience focusing on the future of the domain and delivering value addition to enhance its customer satisfaction.

1.2. ABOUT THE PROJECT

File sharing software's are emerging abundantly nowadays. This has made file sharing one of the top uses of any network. But poor security features is provided by the current soft wares. This Project describes about the design,developement and evaluation of a file sharing system that proposes a novel solution to the shared file security problem. The system will allow users to share files in a secure manner and comprises of client and server applications. The client allows users to connect to a server and share their files amongst all other system users.

The client also gives users the ability to search for files shared by the other system users, and when a file is found it could be transferred from the other user securely, as the file would be encrypted.

The server allows valid users to connect, records their shared file list and enables connected users to search this list. The server authenticates connected users by performing a test that only a valid user can respond to in the correct manner.

The system to be developed uses certain techniques to implement a framework in which to model system security , including authentication of system users and the files that they share. A major objective of the system is to provide a model that can easily scaled with a number of clients. The tests performed show that the response time of the system remains fairly linear to the number of concurrent clients. This proposed project document given below describes the software design specification for the project titled " Software Reliability for Distributed Software Systems"

SCOPE AND OBJECTIVE OF THE PROPOSED SYSTEM

In order to make reliable client server systems, we have to handle application errors. The main aim of this project is to create a system that allows files to be distributed over a network and to be shared among the clients. So that any clients can able to search for other clients that have required files and provide them with a mechanism to share those files.

DEFINITIONS, ACRONYMS AND ABBREVIATIONS
SOFTWARE RELIABILITY

Probability of failure free software operation in a specified environment for a specified Period of time.

CLIENT

A node that makes requests of server in a network or that uses resources available through the servers.

SERVER

A node that provides some type of network service.

CLIENT-SERVER COMPUTING

The main aim of this project is to create a system that allows files to be distributed over a wide area and to be shared in a secure manner. The system should allow clients to search for other clients that have required files and provide them with a mechanism to share those files securely. It involves research in the following subjects:

- Distributed File System Architectures.
- Security Techniques.
- File Transfer Protocols.

REGISTER USER DETAILS

Input: Username, password
Output: Users are registered in the system users table.

Processing

- Client connects to a server and sends the message

New user

- Client sends username and password .
- Server accepts the password and validates the username against with the details stored in the global database.
- Server sends either **NEW_USER_ACK** or **NEW_USER_NAK,** Depending on the result of the validation of the details. If **NEW_USER_NAK** were sent, the server would then send a message to the client, identifying the reason for the rejection. There will be one possible message **USER_EXIST** (username in use) .

Logging In/Out

Input: username, password, shared files
Output: Users are logged on to the server.

Processing

- Client connects to a server and sends the message **EXIST_USER.**
- Client sends username, password and current IP address.
- Server looks up the username from the database and retrieves the password
- Server sends either **LOGON_ACK** or **LOGON_NAK**, depending on the result of the comparison of the received password against the global database password.
- If **LOGON_ACK** were sent to the client, they would then start to send their list of shared files to the server.

- else If **LOGON_NAK** were sent, the server would then send a message to the client, identifying the reason for the rejection. There are two possible messages: **LOGON_PASSWD** (invalid password) or **LOGON_USER** (Invalid username).

File share

Input: file name and file size.
Output: Information of the list of files shared by client are stored in the database

Processing

- Client sends the message **SHARED_START** to the server.
- Client sends the filename and the file size (repeated until all the files in the shared directory are processed).
- Client sends **SHARED_END** to indicate the end of the file list.
- Server attempts to write the shared file list to the local database.
- Server sends either **SHARED_ACK** or **SHARED_NAK**, depending if the shared file list was successfully written to the database or not.

FILE SEARCH

Input: username or file name
Output: information about the client which is having the file will be sent to the requesting client.

Processing

- Client sends the message **FILE_SEARCH** to the server.
- Depending on the search type chosen, the client sends either **FILE_NAME** or **USER_NAME** .

- Server searches its local database and the local databases of the other servers.
- Server sends **FILE_START** to indicate the file search results are being sent.
- Server sends the Filename, File size, Username and User IP Address (repeated until all results from the search are sent).
- Server sends **FILE_END** to indicate the end of the file search results.

FILE TRANSFER

Input: username and file name
Output: File is transferred form target client to the requesting client

Processing

- Requesting client sends the message **FILE_TRANSFER** to the target client.
- Requesting clients sends the filename and username.
- Requesting client can then receive one of two responses: **TRANSFER_ACK** (Filename exists), or **TRANSFER_NAK** (Filename does not exist).
- Target client sends **TRANSFER_START** to indicate the start of the transfer.

ACCOUNT MAINTENANCE

Input: username, old password, new password
Output: new password will be updated in the system users table.

Processing

- Client sends the message **PASSWD_CHNG** to the server.
- Client sends username, old password and new password.
- Server looks up the username from the database and retrieves old password, associated with that username from the global database.
- The received old password is compared against the password from the database.
- Server sends either **PASSWD_ACK** or **PASSWD_NAK**, depending on the result of the comparison of the received old password against the global Database password.
- If **PASSWD_ACK** were sent to the client, their new password would then be written to the global database
- If **PASSWD_NAK** is possible reason for the rejection, so the server sends **LOGON_PASSWD** to inform the client that the incorrect password was supplied.

CHANGE/ UPDATE CLIENT SHARED FILE LIST

Input: new shared file list
Output: shared file list will be updated in the database.

Processing

- Client sends the message **SHARED FILE _CHNG** to the server.
- Client sends the new list of shared files
- Server updates list of new shared files by using steps in the file share module.

CONNECTION ESTABLISHMENT WITH OTHER SERVER

Input: IP address and local and remote port number
Output: Connection is established with the other server.

Processing

- The server will attempt to establish a socket connection to the other servers every five seconds.
- If a connection is established, an ODBC connection is established to the remote server's local database, which is used for client file searches.

CHAPTER 2

SYSTEM REQUIREMENTS AND SPECIFICATION

2.1.HARDWARE REQUIREMENTS SPECIFICATION

System Requirements

Processor	:	Intel Pentium III
RAM	:	256 MB RAM
Hard Disk	:	40GB

Server Requirements

Platform	:	Windows 2000/NT
Database	:	ORACLE

Client Requirements

Working Environment	:	Windows 2000/NT
Runtime Environment	:	JAVA, (JAVA Technologies)

2.2.SOFTWARE REQUIREMENTS SPECIFICATION

Operating System	Windows 2000/NT
Language	JDK 1.4, JAVA Technologies
Database Server	Oracle

2.3.SOFTWARE OVERVIEW

JAVA FEATURES

The inventors are Java wanted to design a language, which could offer solution to some of the problems encountered in modem programming. They wanted the language to be not only reliable, portable and distributed .

Although the above appears to be a list of buzzwords, they aptly describe the full potential of the language. These features have made Java the first application language of the World Wide Web. Java will also become the primer language for general-purpose stand-alone applications.

COMPILED AND INTERPRETED

Usually a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system. First, Java compiler translates source code into what is known as bytecode instructions.

PLATFORM-INDEPENDENT AND PORTABLE

The most significant contribution of Java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet, which inter connects different kinds of systems worldwide. We can download a Java applet from a remote computer on to our local system via Internet an extension of the user's basic system providing practically unlimited number of accessible applets and applications.

ROBUST AND SECURE

Security becomes an important issue for a language that is used for programming on Internet. Threat of virus and abuse of resources is everything. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

DISTRIBUTED

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data programs. Java applications can open and access remote objects on Internet as easily as they can in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

MULTITHREADING AND INTERACTIVE

Multithreaded means handling multiple tasks simultaneously. Java supports multithreading programs. This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip time download an applet from a distance computer. This feature greatly improves the interactive performance of graphical applications.

DYNAMIC AND EXTENSIBLE

Java is a dynamic language. Java is capable of dynamically linking in new class libraries methods and objects. Java can also determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response.

THE JAVA ENVIRONMENT

Java environment a large number of development tools and hundreds of classes and methods. The development tool are part of the system known as **Java Development Kit** (JDK) and the classes and methods are part of the **Java Standard Library** (JSL).

JAVA DEVELOPMENT KIT

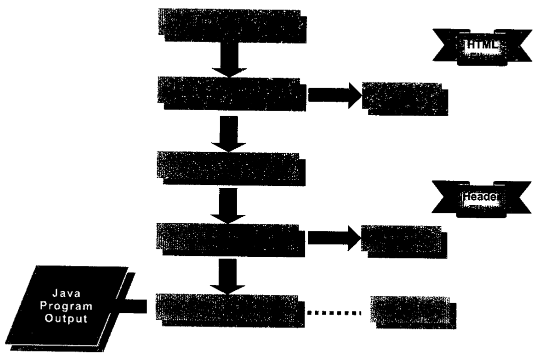
The JDK comes with a collection of tools that are used for developing and running Java program. They include:

➢ AppletViewer	–	for viewing Java applets
➢ javac	–	Java compiler
➢ java	–	Java interpreter
➢ javap	–	Java disassembler

➢ javah	–	for C header files
➢ javadoc	–	for creating HTML documents
➢ jdb	–	Java debugger

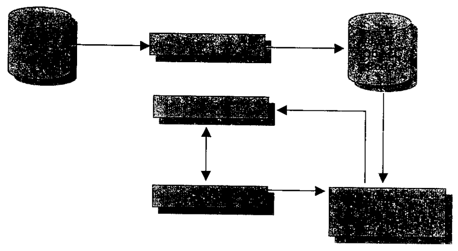
The way these tools are applied to build and run application program is defined in above. To create a Java program, we need to create a source code file using a **text editor.** The source code is then compiled using the Java compiler **javac** and executed using the Java interpreter **java.** The Java debugger **jdb** is used to find errors, if any, in the source code. A compiled Java program can be converted into a source code with the help of Java disassembler **javap.**

THE PROCESS OF BUILDING AND RUNNING JAVA APPLICATION PROGRAMS



THE JAVA VIRTUAL MACHINE (JVM)

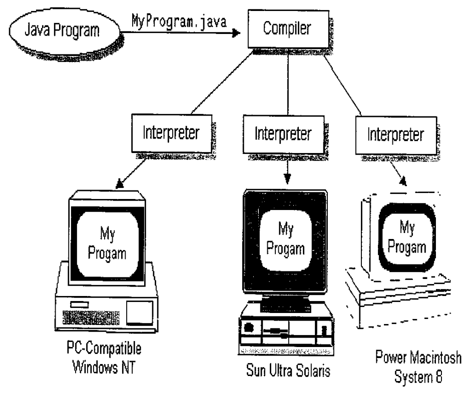
Java Virtual Machine is a Java interpreter form the JavaSoft divisions of Sun Microsystems. It converts the Java intermediate language (bytecode) into machine language a line at a time, and then executes it. The Java Virtual Machine is licensed to software companies that incorporate it into their browsers and server software. Since its is used on all major platforms, Java programs run "virtually" on every computer.



You can think of Java bytecodes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java bytecodes help make "write once, run anywhere" possible. You can compile your Java program into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

JAVA COMPILATION PROCESS



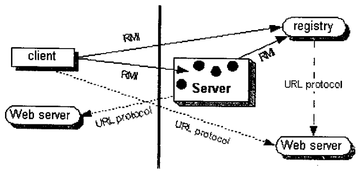
AN OVERVIEW OF RMI APPLICATIONS

RMI applications are often comprised of two separate programs: a server and a client. A typical server application creates some remote objects, makes references to them accessible, and waits for clients to invoke methods on these remote objects. A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a distributed object application. Distributed object applications need to

- Locate remote objects: Applications can use one of two mechanisms to obtain references to remote objects. An application can register its remote objects with RMI's simple naming facility, the rmiregistry, or the application can pass and return remote object references as part of its normal operation.
- Communicate with remote objects: Details of communication between remote objects are handled by RMI; to the programmer, remote communication looks like a standard Java method invocation.
- Load class bytecodes for objects that are passed around: Because RMI allows a caller to pass objects to remote objects, RMI provides the necessary mechanisms for loading an object's code, as well as for transmitting its data.

The following illustration depicts an RMI distributed application that uses the registry to obtain a reference to a remote object. The server calls the registry to associate (or bind) a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method on it. The illustration also shows that the RMI system uses an existing Web

server to load class bytecodes, from server to client and from client to server, for objects when needed



REMOTE INTERFACES, OBJECTS, AND METHODS

Like any other application, a distributed application built using Java RMI is made up of interfaces and classes. The interfaces define methods, and the classes implement the methods defined in the interfaces and, perhaps, define additional methods as well. In a distributed application some of the implementations are assumed to reside in different virtual machines. Objects that have methods that can be called across virtual machines are remote objects.

An object becomes remote by implementing a remote interface, which has the following characteristics.

- A remote interface extends the interface java.rmi.Remote.
- Each method of the interface declares java.rmi.RemoteException in its throws clause, in addition to any application-specific exceptions.

RMI treats a remote object differently from a nonremote object when the object is passed from one virtual machine to another. Rather than making a copy of the implementation object in the receiving virtual machine, RMI

passes a remote stub for a remote object. The stub acts as the local representative, or proxy, for the remote object and basically is, to the caller, the remote reference. The caller invokes a method on the local stub, which is responsible for carrying out the method call on the remote object.

A stub for a remote object implements the same set of remote interfaces that the remote object implements. This allows a stub to be cast to any of the interfaces that the remote object implements. However, this also means that only those methods defined in a remote interface are available to be called in the receiving virtual machine.

CREATING DISTRIBUTED APPLICATIONS USING RMI

When you use RMI to develop a distributed application, you follow these general steps.

1. Design and implement the components of your distributed application.
2. Compile sources and generate stubs.
3. Make classes network accessible.
4. Start the application.

DESIGN AND IMPLEMENT THE APPLICATION COMPONENTS

First, decide on your application architecture and determine which components are local objects and which ones should be remotely accessible. This step includes:

- Defining the remote interfaces: A remote interface specifies the methods that can be invoked remotely by a client. Clients program to remote interfaces, not to the implementation classes of those interfaces. Part of the design of such

interfaces is the determination of any local objects that will be used as parameters and return values for these methods; if any of these interfaces or classes do not yet exist, you need to define them as well.
- Implementing the remote objects: Remote objects must implement one or more remote interfaces. The remote object class may include implementations of other interfaces (either local or remote) and other methods (which are available only locally). If any local classes are to be used as parameters or return values to any of these methods, they must be implemented as well.
- Implementing the clients: Clients that use remote objects can be implemented at any time after the remote interfaces are defined, including after the remote objects have been deployed.

Compile Sources and Generate Stubs

This is a two-step process. In the first step you use the javac compiler to compile the source files, which contain the implementation of the remote interfaces and implementations, the server classes, and the client classes. In the second step you use the rmic compiler to create stubs for the remote objects. RMI uses a remote object's stub class as a proxy in clients so that clients can communicate with a particular remote object.

Make Classes Network Accessible

In this step you make everything--the class files associated with the remote interfaces, stubs, and other classes that need to be downloaded to clients--accessible via a Web server.

Start the Application

- Starting the application includes running the RMI remote object registry, the server, and the client.
- The rest of this lesson walks through the steps to create a compute engine.

Building a Generic Compute Engine

This trail focuses on a simple yet powerful distributed application called a compute engine. The compute engine, a remote object in the server, takes tasks from clients, runs them, and returns any results. The tasks are run on the machine where the server is running. This sort of distributed application could allow a number of client machines to make use of a particularly powerful machine or one that has specialized hardware.

The novel aspect of the compute engine is that the tasks it runs do not need to be defined when the compute engine is written. New kinds of tasks can be created at any time and then given to the compute engine to be run. All that is required of a task is that its class implement a particular interface. Such a task can be submitted to the compute engine and run, even if the class that defines that task was written long after the compute engine was written and started. The code needed to accomplish the task can be downloaded by the RMI system to the compute engine, and then the engine runs the task, using the resources on the machine on which the compute engine is running.

The ability to perform arbitrary tasks is enabled by the dynamic nature of the Java platform, which is extended to the network by RMI. RMI dynamically loads the task code into the compute engine's Java virtual machine and runs the task without prior knowledge of the class that implements the task. An application like this, which has the ability to download code dynamically, is often called a behavior-based application. Such applications usually require full agent-enabled infrastructures. With RMI

such applications are part of the basic mechanisms for distributed computing on the Java platform.

RMI SERVER

The compute engine server accepts tasks from clients, runs the tasks, and returns any results. The server is comprised of an interface and a class. The interface provides the definition for the methods that can be called from the client. Essentially the interface defines the client's view of the remote object. The class provides the implementation.

DESIGNING A REMOTE INTERFACE

This section shows you the `Compute` interface, which provides the connection between the client and the server. You will also learn about the RMI API, which supports this communication.

IMPLEMENTING A REMOTE INTERFACE

This section explores the class that implements the `Compute` interface, thereby implementing a remote object. This class also provides the rest of the code that makes up the server program: a `main` method that creates an instance of the remote object, registers it with the naming facility, and sets up a security manager.

DESIGNING A REMOTE INTERFACE

At the heart of the compute engine is a protocol that allows jobs to be submitted to the compute engine, the compute engine to run those jobs, and the results of the job to be returned to the client. This protocol is expressed in interfaces supported by the compute engine and by the objects that are submitted to the compute engine.

Each of the interfaces contains a single method. The compute engine's interface, `Compute`, allows jobs to be submitted to the engine; the client interface, `Task`, defines how the compute

ORACLE

ORACLE is the most popular open source database server in existence. We started out with the intention of using the mSQL database system to connect to tables using fast low-level (ISAM). ORACLE is a relational database. It ensures that transactions comply with the ACID model, allows the building of indexes, supports standard data types, and allows for database replication, among other features. The architecture of ORACLE makes it fast, reliable and easy to use along with having all the capabilities required to handle corporate database applications. Handles large databases, in the area of 50,000,000+ records. No memory leaks.

Features of ORACLE

The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications. Such applications can be used in isolation or in environments where no network is available.The RDBMS is distinguished by three characteristics

- A very fast thread-based memory allocation system.
- Provides transactional and non-transactional storage engines.
- Connectability.

ORACLE Compatibility

ORACLE is a database programming language. Modern ORACLE has evolved into a widely used standard for relational databases. The ORACLE language is composed of commands, clauses, operators and aggregate functions. These elements are combined into statements to create, update and manipulate

databases. ORACLE is completely non-procedural with no complex programming or complex loops. One has to simply tell the system what he wants. It is the common language for all relational databases. That is portable and it requires only small modification to make use of in the other databases. It is nonprocedural language because more than one record can be accessed rather than one record at a time.

ORACLE (DLL)

It includes number of commands to create tables and indexes, and modify them by adding or removing columns and indexes.

ORACLE (DML)

The ORACLE Data Manipulation Language statements are used to perform operations on the data in the tables. There are ORACLE statements for selecting records from a table, for adding new records, updating or deleting them.

Portability

ORACLE runs on a wide range of main frames, minis, and personal computers. Oracle is available on virtually all-major systems. With ORACLE one can develop an application program in one environment and execute the same application without any modifications in an other environment running oracle. The data itself is transportable.

Connectability

Having the same software running on different machines simplifies the task of connecting the machines to network. ORACLE is networking software allows the users and applications to directly access data stored in a shared database.

Windows 2000/NT

This is a network operating system. This is based on the client-Server architecture. Benefits of this Operating system is as follows

- More intuitive interface.
- Better multitasking and multithreading.
- Clients can be attached to workstations.
- Plug-and-Play technology.
- Higher level of security.
- NTFS-a powerful NT File System.

CHAPTER 3

SYSTEM ANALYSIS

INTRODUCTION

System analysis is concerned with investigating and analyzing which is used to gain an understanding of the existing system and what is required. It is a general form refers to orderly structured process for identifying and problem solving.

System analysis is the application of the systems approach to problem solving using computers. The ingredients are systems elements, processes, and technology. This means that to do systems work, one needs to understand the systems concept and how organizations operate as a system.

System analysis is a process related to four significant phases namely study phase, design phase, development phase and operation phase. The definition of the system analysis is not only the process of analysis but also that of synthesis. System analysis is actually a customized approach to the use of the computer for solving problem.

Many existing technologies allow clients to share files amongst themselves. The main improvement this system offers over the other available a technology is that the files will be shared securely. The security aspects will involve encryption of the file before it is sent and subsequent decryption by the recipient of the file. This means only the intended recipient will be able to read the contents of the file, so if the data transmission were intercepted by a third Party, it would be undecipherable.

Another security measure of the system should allow it to detect invalid users. Therefore, if a hacker were able to break into a communication stream, the system would be able to validate the authenticity of the user and then take appropriate action.

CHAPTER 4

SYSTEM DESIGN

INTRODUCTION

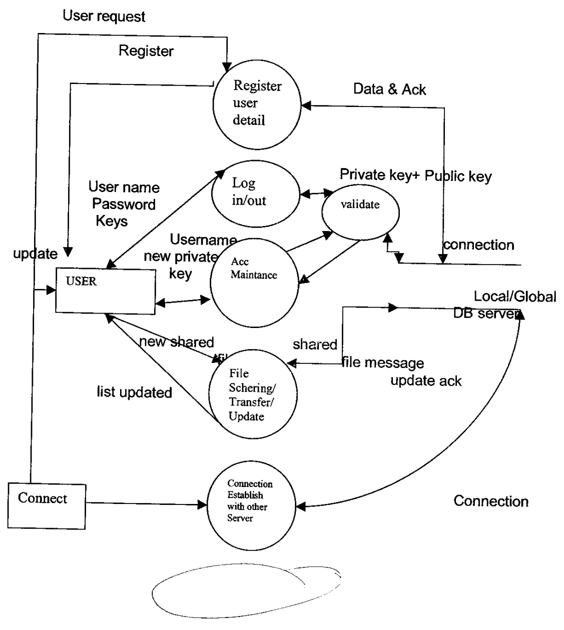
The original thinking behind this system was it could be deployed in a commercial environment, where a group of users who require the need to share files amongst themselves in a secure manner could subscribe to the service, and guarantee their authenticity. Therefore, if it were to be deployed in a business environment, only authorized users would be able to gain access share files with each other.

File sharing can already be achieved with a standard network operating system, as access permissions can be set to directories, giving access only to authorised users. However, this is not the safest or most efficient method of sharing files as passwords can be obtained and transmitting files over a wide area using a network can be slow.

4.1.DATA FLOW DIAGRAM

Data flow diagrams are commonly used during problem analysis and design. A DFD shows the flow of data through a system. It views the system as a function that transforms the inputs into desirable outputs. A DFD aims to capture the transformation that takes place within a system into output data so that eventually the output data is produced.

The agent that performs the transformation of data from one state to another is called a process (Bubble). The processes are shown by named circles and dataflow are represented by named arrows. A square defines a source or destination of system data. An open rectangle is a data source



4.2 DATABASE DESIGN

4.2.1 Table Name : field detail

#	Screen Label	Field Type	Constraint	Max Len	Mandatory
1.	filename	Varchar2	-	10	Yes
2.	Computername	Varchar2	-	20	Yes
3.	Local IP address	Varchar2	-	10	Yes
4.	size	numeric	-	20	Yes

4.2.2 Table Name : newuserid

#	Screen Label	Field Type	Constraint	Max Len	Mandatory
1	name	Varchar2	-	10	Yes
2	password	Varchar2	-	10	Yes
3	confirmpassword	Varchar2	-	10	Yes

4.2.3 Table Name : server

#	Screen Label	Field Type	Constraint	Max Len	Mandatory
1	servername	Varchar2	-	10	Yes
2	Server ip address	Varchar2	-	20	Yes
3	port	numeric	-	4	Yes

4.3 INPUT AND OUTPUT DESIGN

INPUT DESIGN

Input design is the process of correcting a user-oriented description of the inputs to a computer based one. Inaccurate data is one of the most common causes of data processing errors. If poor input design, particularly where operators enter data from source documents permit wrong data to enter into a computer system, then it will change the entire process in an unpleasant way consists of

- The field length is well documented. The field length is specified correctly such that the data entered will not exceed the allocated space and or numeric data is right justified.
- The sequence of field matches the sequence of data or type of data which is going to be entered.
- The data format is well identified for entering or specifying the data's

In this system, the screen includes appropriate labels or prompts for data entry. The system also includes some screen design rules that are important for user satisfaction. They are given below.

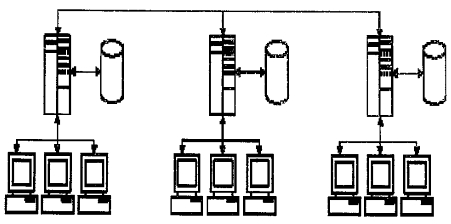
1) The same format is used with related screens, users can identify easily where the selections are made while placing orders.
2) The screen is not over crowded. Often too neat, and eye pleasing. It facilitates the user to identify the labels easily and enter the data's.
3) Provides validation instructions if the user enters a numeric field in a character field.
4) The consistent terminology is used which coordinate forms and screen designs.

5) Hence the input design will be easy to follow and does not induce errors. The screen designs are viewed in appendix.

OUTPUT DESIGN

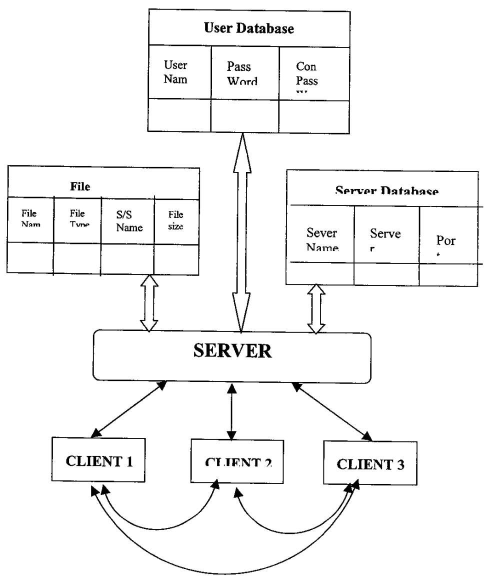
Outputs from a system can be defined as the information being processed and then generated by the system in a specified format. Output design serves the best information source of any system. Once the output is designed it would serve for present and future references. Outputs are carefully designed such that it gives an error free output format.

MULTI SEVER ARCHITECTURE



4.3.2 Figure multi Server Architecture

BLOCK DIAGRAM



4.3.3 FIGURE CLIENT SERVER BLOCK DIAGRAM

FILE SYSTEM ARCHITECTURE

The research conducted into distributed file system architectures outlined that the Client-Server and Gnutella architectures are completely unsuitable for this type of application, as both architectures do not meet the system requirements. In the Client-Server architecture, files would have to be placed on the central server by the clients in order to be shared. This does not meet the system requirements, as only the client should host the file and should only release it when they want to. If the file was to reside on a server, it could be viewed by any user with enough access rights. In the Gnutella architecture, there is no central server to store user information as clients connect directly to one another.

This does not meet the system requirements either, as there would be no method to validate the authenticity of a client. The remaining Napster architecture will be used for the system model. This architecture meets the requirements of the system, as the central server can store user information to validate client authenticity. In addition, files are shared between the clients, as the central server offers connected clients the facility to search for a file and provides the address of the client hosting that file.

SYSTEM MODEL

The Napster architecture was used as a basis for system design, out of which, three system model candidates were developed:

Single Server

This model consists of one server and when a client successfully connects, their session details and files they are sharing are recorded in the database. When connected, the client is free to interrogate the server for any required files, with the connection details of clients that hold those files returned. Unfortunately, this model

suffers from a serious problem, as there is a central point of failure for the system. So if the central server was to become unavailable, so would the entire system. This could be the result of a hacker launching a denial of service attack. This bombards the server with requests that slows it down to the point where no one can gain access.

Multiple Server

This model consists of three servers and when a client successfully connects to either of them, their session details and files they are sharing are recorded in the server database. This information is then replicated to the other two servers, so each server maintains a global list of connected clients and shared files. This gives a client connected to Server one the ability to search and commence download of a file hosted by a client connected to Server 2 or Server 3. Unfortunately, this model suffers from a problem that may lead to database inconsistencies. This would occur if one of the servers were to become unavailable, as the databases in the functional servers would indicate that clients are still connected to the unavailable server and that their files are still available.

Multiple Server with Controlling Server

This model consists of four servers, with one used exclusively for administering replication between the three remaining servers. This solves the database inconsistency problem in the multiple server model, as functional servers are updated immediately when the controlling server detects that a server is unavailable. In this model, when a client connects to any server, their session details and files shared are recorded in the local server database, which is replicated to the controlling server and finally replicated to the remaining two servers.

Unfortunately, the multiple server model suffers from the same problem as the single server, as there is a central point of failure for the system, although if the controlling server was to become unavailable, the three remaining servers would be able to provide a partial service to their locally connected clients.Out of these three possible system models, the model that will be used to implement the system is the Multiple Server model, is The reason behind this decision is there is no central point of failure for the system, which the other two models suffer from. The database inconsistency problem that this model suffers from can be resolved using WinSock programming, as each server will maintain a socket connection to the other two servers, and will immediately be able to detect if a server disconnects, due to the connection-oriented nature of sockets.

SYSTEM SECURITY

The Theory chapter of this report outlined the three main cryptographic techniques used for the encryption of files. This section discusses the suitability of these three techniques in context of the system. In secret-key cryptography, the sender and receiver must have the same key to encrypt and decrypt the file. This cryptographic technique is not suitable for the system as the key would have to be sent from sender to recipient with every file transfer, or every system user would have to have the same key. In either case, the security of the key is not guaranteed. In public-key cryptography, each user has two related keys, a public-key and a private key. The private key is only known by owner of the key and is used to decrypt messages encrypted by the corresponding public-key, which is available to anyone.

CLIENT DESIGN

This section identifies the operations and functionality of the client application, illustrating how the client application interacts with the user and the server application. The Client design is illustrated in Appendix 1.

Register User Details

When the client application is run for the first time, it will check if the user has previously registered, possibly using the windows registry. If no previous user details have been detected, the application will prompt the user to register their details (username, password, public-key and modulus key).
Once the details are entered, the application randomly selects one of the servers, connects and informs the server that a new user is registering their details. It then sends the user details to the server. The Server then validates these details and informs the user if they have been accepted or rejected, if they are rejected, the server informs the user of the reason for rejection. The user may be rejected if the username or the public/modulus keys are already in use, in either case, the user can choose a new username or regenerate the public/private key combination.

Logging In/Out

When a registered user attempts to login, the client application selects one of the servers at random and connects. The server authenticates their details, either granting or denying them system access. If they are denied access, the server informs the user of the reason for rejection, which may be an invalid username or password. If they are granted access, the list of files the user is sharing is sent to the server and made available to all system users. When the user logs out, the server deletes the shared file entries for that user from the database.

Account Maintenance

When logged on, a user will have the ability to change their password or their public/private key combination. In both cases, the user will submit the new details to the server for validation. The server will then inform the user if the change was successful or not.

File Searching/Transferring/Updating

When logged on, a user can submit search queries by filename or username to the server, which will return any matches. The user can select any of these files and commence file transfer by establishing a connection to the user who hosts the file. The file is encrypted by the user hosting the file using a randomly generated session key, which is encrypted itself using the public-key of the requesting user. Finally, both the encrypted file and the encrypted session key are then sent. At any time, a connected user can update their list of files they are sharing, which happens automatically when the user first logs on to the server. If the user is already logged on, the server deletes the shared file entries for that user from the database and writes the new entries submitted by the user.

SERVER DESIGN

This section identifies the operations and functionality of the server application, illustrating how the server application interacts with the client and the other servers.The server design is Illustrated in Appendix 2, and has the following parts:

Registering Client Information

When a new user connects to the server, they will supply their username, encrypted password and public/modulus keys. To ensure the username and public/modulus keys are unique, the server checks them against the existing entries in the database, and notifies the client if they have been registered successfully or not. If the registration was successful, the new user details are written to the database and immediately replicated to the other server databases.

Validating Clients

When an existing client connects, they supply their username,encrypted password and their current IP address. The server looks up the username from the database and retrieves the password and public-key associated with that username. The received password is decrypted, compared against the password from the database and the user is informed of the result. If the logon attempt was successful, the username and current IP address of the user is written to the server database, so the other system users can connect directly via the IP address.

Recording Shared Files

When a user logs on to a server, they automatically send the server a list of files they are sharing. The server writes this file list into the database, making the files available to all other system users. While connected, a user can update their shared file list with the server any time.

Searching for Files

When a connected user submits a file search to the server, it first queries its own local database for any files matching the search criteria. If there are any active connections to the other servers, the search query is then sent to their databases via the ODBC connection. Finally, any results from the file search are sent to the user.

Updating Client Passwords

When a connected user requests to change their password, they send their username, encrypted old password and encrypted new password. The server looks up the username from the database and retrieves the password and public/modulus keys associated with that username. The received old password is decrypted, compared against the password from the database and if they match, the database is updated with the new password.

Updating Client Keys

When a connected user requests to change their keys, they end their username, encrypted password and new public/modulus keys. The received password is decrypted using the old key values from the database and compared against the password from the database. If they match, the server checks the database for the new public/modulus key combination, in order to determine if they are currently in use. If they are not in use, the new keys are written to the database.

OTHER SERVER CONNECTIONS

When a server is started, the administrator can attempt to connect to the other servers manually, or allow the server to attempt to connect automatically. If the automatic connection is chosen, the server will attempt to establish a socket connection to the other servers every five seconds. When a socket is not attempting to connect to a server, it will listen for a connection. This way if all servers in the system are set connect automatically, they will all do so within a relatively small amount of time.

When a socket connection is established from one server to another, an ODBC connection is also created. Now every file search submitted by clients connected to either of these servers is queried first in the local database of the server, followed by the other server's remote database. If a server were to become unavailable, the servers connected to it would detect this using their socket connections. The servers would then close their ODBC connections to the unavailable server and only submit file searches to their local databases and the remote databases of the remaining connected servers.

Server sends either **LOGON_ACK** or **LOGON_NAK**, depending on the result of the comparison of the received password against the global database password. If **LOGON_ACK** were sent to the client, they would then start to send their list of shared files to the server using the Shared File List Protocol. If **LOGON_NAK** were sent, the server would then send a message to the client, identifying the reason for the rejection. There are two possible messages: **LOGON_PASSWD** (invalid password) or **LOGON_USER** (Invalid username).

File Search Method

This protocol is used when an existing user is connected to a server and they submit a file search to the server.

- Client sends the message **FILE_SEARCH** to the server.

- Depending on the search type chosen, the client sends either **FILE_NAME** (followed by the filename) or **USER_NAME** (followed by the username).
- Server searches its local database and the local databases of the other servers.
- Server sends **FILE_START** to indicate the file search results are being sent.
- Server sends the Filename, File size, Username and User IP Address (repeated until all results from the search are sent).
- Server sends **FILE_END** to indicate the end of the file search results.

CHAPTER-5

IMPLEMENTATION

5.1 SYSTEM TESTING

System testing makes a logical assumption that if all the part of the system is correct and the goal will be successfully achieved. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live commences.

There should be careful planning how the system will be proved and the test data designed. During system testing, the system is used experimentally to ensure that the software does not fail. In other words, we can say that it will run according to its specifications and in the way users expect.

It is desirable to discover any surprises before the organization implements the system and depends on it. Software testing accounts for the largest percentage of technical effort in the software process. Yet we are beginning to understand the subtleties of systematic test planning, execution and control. The objective of system testing is to uncover errors. To fulfill the objective the system testing is done in the following three phases.

Unit Testing

Unit testing focuses verification effort on the smallest unit of the software design the module. The local data structure is examined to ensure that data stored temporarily maintains its integrity. Boundary conditions are tested to uncover the errors of the module within the boundary.

Integration Testing

After the complete testing of all modules, they are put together and integrated. The primary concern is the compatibility of individual modules. The specification for data type, length and name in each module is also tested for compatibility.

Stress Testing

Once all the modules are subjected to stress test, to confront programs with abnormal situations. The various stress tests may be based on choosing large tables with large number of fields. Several users were allowed to access the database simultaneously and the system survived this test successfully.

All the modules of this system were successfully tested using test data as well as real data collected. All the reports and the screens are tested for their validity and values in the database tables are checked for their correctness and consistency. After successful testing of the system, it was ready for implementation.

5.2 MAINTENANCE

The process of making changes and modifications to the system after it has been delivered implemented and is in use called software maintenance.

Corrective Maintenance

It is concerned with fixing reported errors in software. They are coding errors and design errors.

Adaptive Maintenance

It is concerned with changing the software to source and to adapt to the new and changing environment.

Defective Maintenance

It involves implementing new functioned or non-functional system requirements to ensure more effective execution of the system.

Perceptive Maintenance

It mainly deals with accommodating new or changed users requirements. It also includes activities to increase the system performance or to enhance its user interface. The objective of perceptive maintenance should be to prevent failures and optimize the software.

Preventive Maintenance

It concerns activities aimed at increasing the system's maintainability such as updating documentation adding comments, improving modular structure of the system.

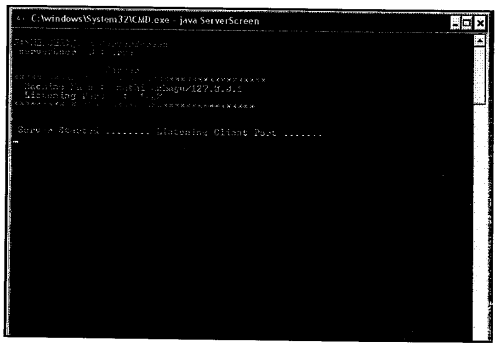
CHAPTER 6

CONCLUSION

To design ,develop and evaluate a file sharing system. Here we provide solution to shared file security problems. We allow user to share files in a secure manner. It comprises client server applications. To provide a model that can be easily scaled with a number of clients.

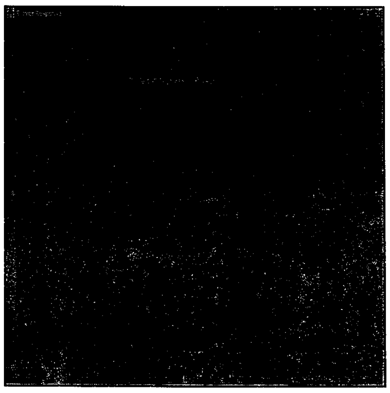
APPENDICES

APPENDIX-1

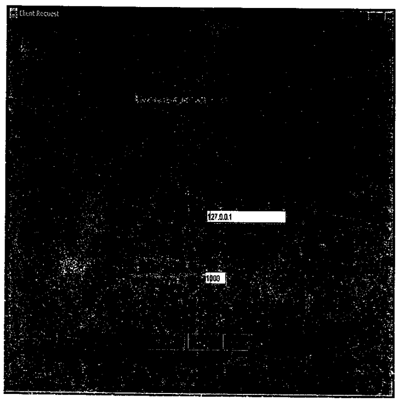


```
[illegible]
[illegible]

          [illegible]
[illegible]
  Machine Name : [illegible]/127.0.0.1
  Listening Port : [illegible]
[illegible]

Server Started ........ Listening Client Port .......
```

RMISSERVER



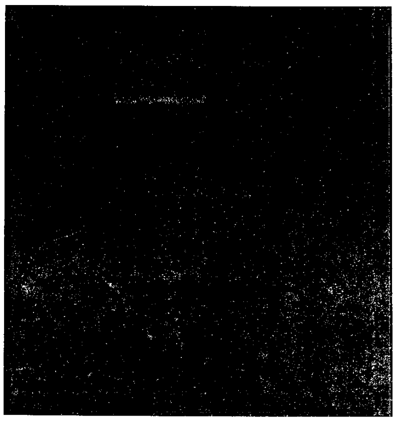
SERVER SCREEN



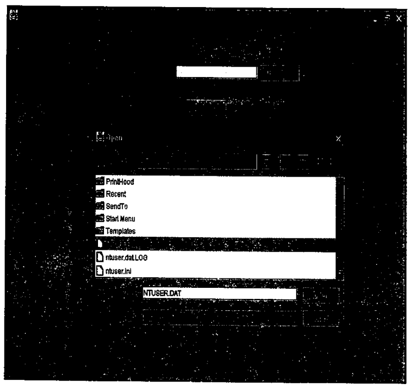
127.0.0.1

1000

CLIENT SCREEN



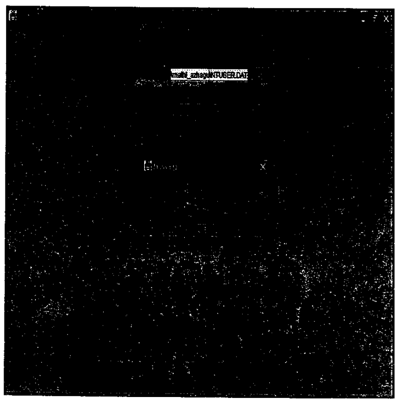
NEW USER LOGIN 1



PrintHood
Recent
SendTo
Start Menu
Templates
ntuser.dat.LOG
ntuser.ini

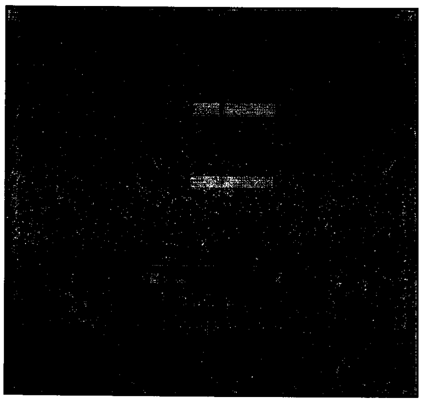
NTUSER.DAT

BROWSING A FILE

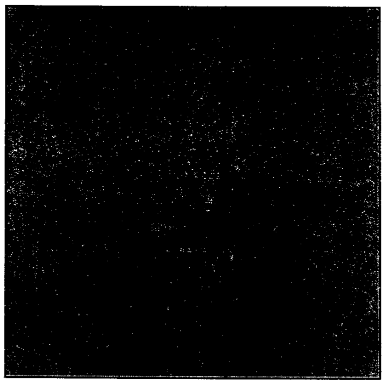


malthi_azhagu\NTUSER.DAT

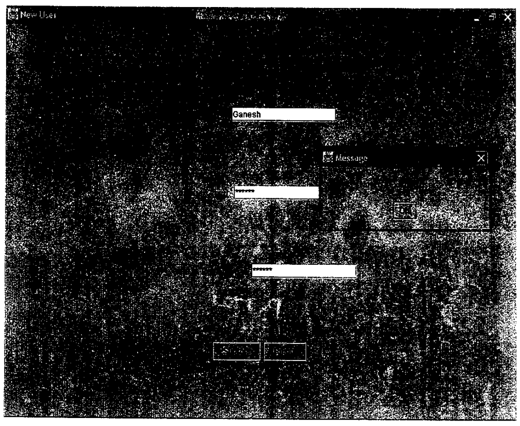
SHARING A FILE



USER LOGIN 2



REQUESTING A FILE



Register New User

REFERENCES

BOOKS

1. Herbert Scgildt (1995) Complete Reference
2. Peter ~~Nortons~~ (1998) Complete Guide To Networking
3. Behrouz A.Forouzan (2000) Netwoking

WEBSITES

1. www.rmi tech.com

2. www.sun java.com.

3. www.mhhe/forouzan.com.