P-3614

DYNAMIC SEARCH ALGORITHM IN UNSTRUCTURED PEER-TO-PEER NETWORKS

PROJECT REPORT

Submitted by

MADAMANCHI PHANI KUMAR Reg.No: 0710108026

G.NAVIN KUMAR Reg.No: 0710108031

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE - 641049

Department of Computer Science and Engineering

PROJECT WORK, April 2011

This is to certify that the project entitled "**DYNAMIC SEARCH ALGORITHM IN UNSTRUCTURED PEER-TO-PEER NETWORKS**" is the bonafide work of **MADAMANCHI PHANI KUMAR, G.NAVIN KUMAR,** who carried out the project work under my supervision.

[Mr.T.Sudhakar M.E.,]
Project Guide

[Mrs.P.Devaki M.E.,(Ph.D),]
Head of the Department/CSE

The candidates with **university Register Nos. 0710108026 & 0710108031** was examined by us in project viva-voce examination held on 20.04.2011.

Internal Examiner

External Examiner

DECLARATION

We,

MADAMANCHI PHANI KUMAR **Reg.No: 0710108026**
G.NAVIN KUMAR **Reg.No: 0710108031**

Hereby declare that the project entitled **"DYNAMIC SEARCH ALGORITHM IN UNSTRUCTURED PEER-TO-PEER NETWORKS"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Engineering (Computer Science and Engineering) degree, is record of original work done by us under the supervision and guidance of Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore
Date: 19.04.2011

[Madamanchi Phani Kumar] [G.Navin Kumar]

Project Guided by,

[Mr.T.Sudhakar M.E.,]

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.,** for giving this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran, M.Tech., (Ph.D).**, *Principal* for providing the necessary facilities to complete my project work.

I take this opportunity to thank **Dr.S.Thangasamy Ph.D.,** *Dean, Research and Development,* for his precious suggestions. I also thank **Mrs.P.Devaki M.E., (Ph.D),** *HOD,* Department of Computer Science and Engineering, for her support and timely motivation.

I register my hearty gratitude to our Guide **Mr.T.Sudhakar M.E.,** *Assistant Professor,* Department of Computer Science and Engineering. I thank for his support, encouragement and ideas and also I thank him for the countless hours he has spent with us for discussing various platforms of research to academic choices.

I would like to convey my honest thanks to all **Teaching** and **Non Teaching** staff of the department for their support. I would like to thank all my classmates who gave me a proper light moments and study breaks apart from extending some technical support whenever I needed them most.

CONTENTS

ABSTRACT

Designing efficient search algorithms is a key challenge in unstructured peer-to-peer networks. Flooding and random walk (RW) are two typical search algorithms.

Flooding searches aggressively and covers the most nodes. However, it generates a large amount of query messages and, thus, does not scale. On the contrary, RW searches conservatively. It only generates a fixed amount of query messages at each hop but would take longer search time.

We propose the dynamic search (DS) algorithm, which is a generalization of flooding and RW. DS takes advantage of various contexts under which each previous search algorithm performs well. It resembles flooding for short-term search and RW for long-term search. Moreover, DS could be further combined with knowledge-based search mechanisms to improve the search performance. Numerical results show that DS provides a good tradeoff between search performance and cost.

LIST OF FIGURES

LIST OF ABBREVATIONS

RW : RANDOM WALK

DS : DYNAMIC SEARCH

MBFS: MOST BREADTH FIRST SEARCH

P2P : PEER-TO-PEER

BFS : BREADTH FIRST SEARCH

DFS : DEPTH FIRST SEARCH

TTL : TIME-TO-LIVE

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

In unstructured peer-to-peer (P2P) networks, each node does not have global information about the whole topology and the location of queried resources. Because of the dynamic property of unstructured P2P networks, correctly capturing global behavior is also difficult. Search algorithms provide the capabilities to locate the queried resources and to route the message to the target node. Thus, the efficiency of search algorithms is critical to the performance of unstructured P2P networks

1.2 EXISTING SYSTEM

Previous works about search algorithms in unstructured P2P networks can be classified into two categories: breadth first search (BFS)-based methods, and depth first search (DFS)-based methods. These two types of search algorithms tend to be inefficient, either generating too much load on the system or not meeting users' requirements. Flooding, which belongs to BFS-based methods, is the default search algorithm for Gnutella network. By this method, the query source sends its query messages to all of its neighbors. When a node receives a query message, it first checks if it has the queried resource. If yes, it sends a response back to the query source to indicate a query hit. Otherwise, it sends the query messages to all of its neighbors, except for the one the query message comes from. On the other hand, random walk (RW) is a conservative search algorithm, which belongs to DFS-based methods. By RW, the query source just sends one query message (walker) to one of its neighbors. If this neighbor does not own the queried resource, it keeps on sending the walker to one of its neighbors, except for the one the query message comes from, and thus, the search cost is reduced.

1.3 PROBLEM IDENTIFICATION

An existing system approaches, the main drawback of RW is the long search time. Since RW only visits one node for each hop, the coverage of RW grows linearly with hop counts,

which is slow compared with the exponential growth of the coverage of flooding. Moreover, the success rate of each query by RW is also low due to the same coverage issue. Increasing the number of walkers might help improve the search time and success rate, but the effect is limited due to the link degree and redundant path. The drawback of flooding is the search cost. It produces considerable query messages even when the resource distribution is scarce. The search is especially inefficient when the target is far from the query source because the number of query messages would grow exponentially with the hop counts.

1.4 PROPOSED SYSTEM

In this paper, we propose the dynamic search (DS) algorithm, which is a generalization of flooding and RW. DS overcomes the disadvantages of flooding and RW and takes advantage of different contexts under which each search algorithm performs well. The operation of DS resembles flooding for the short-term search and RW for the long-term search. In order to analyze the performance of DS, we apply the random graphs as the models of network topologies and adopt the probability generating functions to model the link degree distribution. We evaluate the performance of search algorithms in accordance with some performance metrics including the success rate, search time, number of query hits, number of query messages, query efficiency, and search efficiency.

Simulation experiments are performed in a dynamic P2P networking environment in order to collect convincing results for algorithm evaluations. The factors considered include the network topology, link degree distribution, peer's joining and leaving, and querying behavior as well as the activity of file sharing. Our dynamic network model is constructed based on these factors that strongly reflect the real measurement studies. Numerical results show that DS could provide a good tradeoff between search performance and cost. On average, DS performs about 25 times better than flooding and 58 times better than RW in power-law graphs, and about 186 times better than flooding and 120 times better than RW in bimodal topologies.

CHAPTER 2

SYSTEM DESIGN

Design is a multi-step process that focuses on data structure, software architecture, procedural details, (algorithms etc...) and interface between the modules. The design process also translates the requirements into the representation of software that can be accessed for quality before coding begins.

Computer software design changes continually as new methods; better analysis and broader understanding evolve. Software design is at a relatively early stage in its revolution, software design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However techniques for software designs do exist, criteria for design qualities are available and design notations can be applied.

Once software requirements have been analyzed and specified, software design is the first of three technical activities- Design, code and test that we required to build and verify software. Each activity transforms information in a manner that ultimately results in validation of the computer software.

The importance of software design can be started with a single word quality. Design is the place where quality fostered in software development. Design provides us with the representation of the software that can be accessed for quality.

Design is the only way that can accurately translate a customer's requirements into a finished software product or system. Without design, risk of building an unstable system exists – one that will fail when small changes are made one that may be difficult to handle.

2.1 INPUT DESIGN

A process of converting user originated inputs to computer-based format. Input design is in an important part of development process since inaccurate input data are the most common cause of errors in data processing, erroneous entries can be controlled by input design.

It consists of developing specifications and procedures for entering data into a system and must be in a simple format. the goal of input data design is to make data entry as easy, logical and free from errors as possible. In input data design, we design the source document that capture the data and then select the media used to enter them in to the computer.

2.2 OUTPUT DESIGN

Computer output is the most important and direct source of information to the user. efficient output design should improve the system relationship with the user and help in decision making. the task output preparation is critical requiring skill and ability to align user requirements with capabilities of the system in operation.

- The standard for the printed output suggest the following.
- Give each output a specific name or title.
- Provide a sample of the output layout including areas where printing may appear and location of each field.
- State where each output field is to include significance zeroes, spaces between fields, alphabetic or any other areas.
- Specify the procedure for providing the accuracy of output data.

CHAPTER 3
PROJECT DESCRIPTION

3.1 MODULES

- Peer Construction
- Search process
- Updating History table

PEER CONSTRUCTION

In this module we design a loosely connected P2P network. We use **P** to denote a single peer in the network. The main type of resource, namely, a data file, is represented by f.

SEARCH PROCESS

In this module we use guided search protocol to searching the files in the peer to peer network. In case that the peer chosen to forward the query already exists in the search history **hq**, another peer with a relatively high probability of satisfying the query will be chosen instead

UPDATING HISTORY TABLE

In this module the routing table consists of peer name, file location, nc of times visited information's. Entry such as peer name, file location, and increase the file count value of number of times file visited history.

3.2 IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned Into a working system and is giving confidence on the new system for the users, which it will work efficiently and effectively. It involves careful planning, investigation of the current System and its constraints on implementation, design of methods to achieve the change over, an evaluation, of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation.

An implementation co-ordination committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

Implementation is the final and important phase, the most critical stage in achieving a successful new system and in giving the users confidence. That the new system will work be effective .The system can be implemented only after through testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

3.3 DIAGRAMS

3.3.1 DATA FLOW DIAGRAM

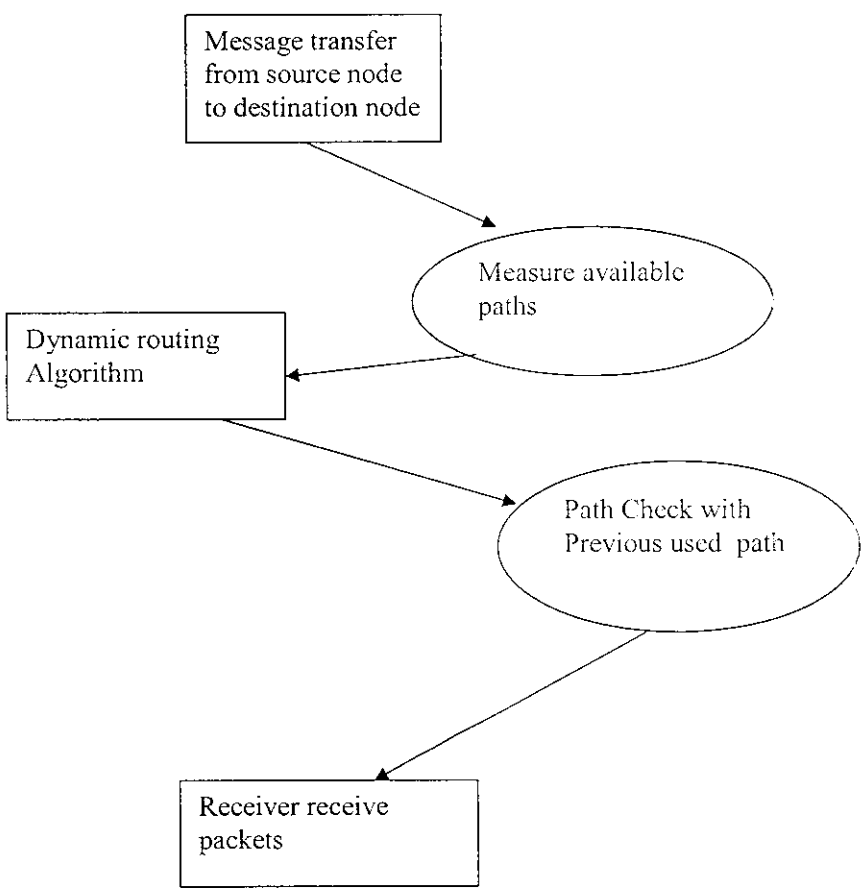


Fig: 3.3.1 data flow diagram for P2P communication

3.3.2ARCHITECTURE DIAGRAM

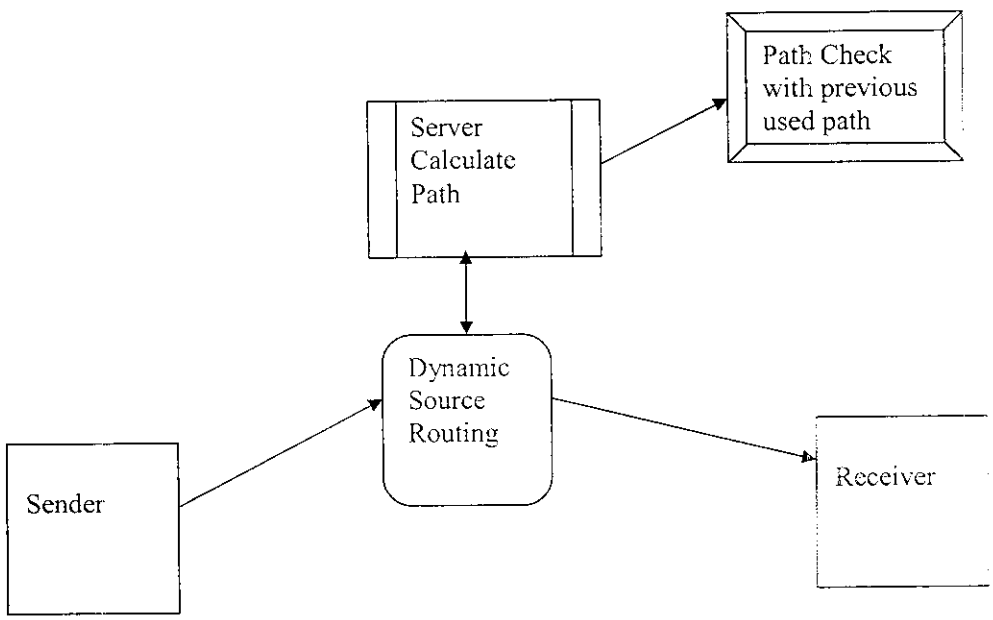


Fig : 3.3.2 architecture diagram for P2P communication

3.3.3 ARCHITECTURE OVERVIEW

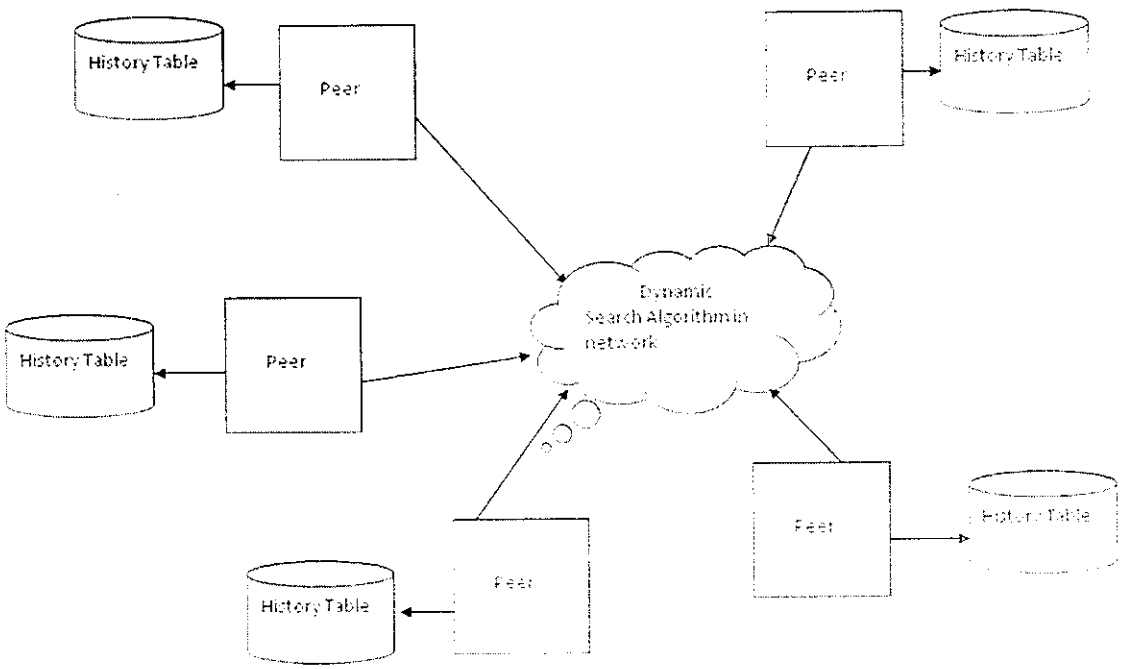


Fig 3.3.3 architecture overview for peer – peer communication

3.4 DATABASE DESIGN

ROUTING TABLE

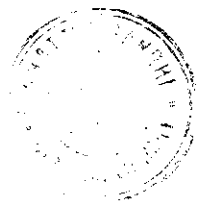
Name	Type	Size
NodeName	VarChar	10
File Name	VarChar	30
System Name	VarChar	30

Tab 3.4.1: routing table

PEER CONSTRUCTION TABLE

Name	Type	Size
Peer no	VarChar	50
IP	VarChar	50
Port	Numeric	9(18,0)
Status	VarChar	50

Tab 3.4.2: peer construction for constructing peer



3.5.1 ALGORITHM DESCRIPTION

DS is designed as a generalization of flooding, MBFS, and RW. There are two phases in DS. Each phase has a different searching strategy. The choice of search strategy at each phase depends on the relationship between the hop count h of query messages and the decision threshold n of DS.

3.1.1 Phase 1. When h < n

At this phase, DS acts as flooding or MBFS. The number of neighbors that a query source sends the query messages to depends on the predefined transmission probability p. If the link degree of this query source is d, it would only send the query messages to d _ p neighbors. When p is equal to 1, DS resembles flooding. Otherwise, it operates as MBFS with the transmission probability p.

3.1.2 Phase 2. When h > n

At this phase, the search strategy switches to RW. Each node that receives the query message would send the query message to one of its neighbors if it does not have the queried resource. Assume that the number of nodes visited by DS at hop h ¼ n is the coverage cn, and then the operation of DS at that time can be regarded as RW with cn walkers. However, there are some differences between DS and RW when we consider the whole operation. Consider the simple scenario shown in Fig. 1. Assume that the decision threshold n is set as 2. When h > 2, DS performs the same as RW with c2 ¼ 12 walkers. Let us consider an RW search with K ¼ 12 walkers. At the first hop, the walkers only visit four nodes, but the cost is 12 messages. RW would generate a large amount of redundant messages when K is set too large. Suppose that s is the query source, r is the vertex that receives the query message, f is the queried resource, mi is the ith query message, and TTL is the time-to-live limitation. Fig. 2 shows the pseudocode of DS. In short, DS is designed to perform aggressively for the short-term search and conservatively for the long-term search. Obviously, the parameters n and p would affect the performance of DS. In Section 3.2, we will analyze the performance ofDSandshowthe effects of parametersnand p.

CHAPTER 4
SYSTEM CONFIGURATION

System Configuration provides the information about the components required to implement the project. The component includes both hardware and software.

4.1 HARDWARE CONFIGURATION

Processor	:	Pentium IV
Speed	:	500 MHz
Main Memory	:	512 RAM
Hard Disk	:	40 GB
Keyboard	:	104 Keys Keyboard
Mouse	:	Optical Mouse

4.2 SOFTWARE CONFIGURATION

Operating System	: Windows XP
Development tool	: EditPlus
Language	: Java 1.6
Front end	: swing
Database	: sql server 2000

CHAPTER 5

SYSTEM ANALYSIS

5.1 FEASIBILITY ANALYSIS

Feasibility studies are preliminary studies during project development that are typically used to demonstrate proof of principle. These studies are exempt from the testing turn around requirements, as well as any requirements associated with studies that would be used part of a submission.

The system has been tested for feasibility in the following points.

- Technical feasibility
- Economical feasibility
- Operational feasibility

5.1.1 TECHNICAL FEASIBILITY

The project entitles "**Dynamic Routing with Security Considerations**" is technically feasibility because of the below mentioned feature. The project was developed in java with sql server database. It provides a high level of reliability availability and compatibility. All these java an appropriate for this project.

5.1.2 ECONOMICAL FEASIBILITY

The entitles "**Dynamic Routing with Security Considerations**" will help in automated searching of the online e-learning process, which leads the accurate finding of suitable checklists and the compliances in the product. With this project we can search the learning object in online e-learning process.

5.1.3 OPERATIONAL FEASIBILITY

In this project the administrator know the details of process in the organization and the data will be maintained as centralized and if any inquires for that particular product can be known as per their requirements and necessaries people are inherently to change and computers have been known to facilitate change an estimate should be made strong a reaction the user staff is likely to have towards the development of a new system. The employees arte accustomed to proposed system.

CHAPTER 6

SYSTEM STUDY

6.1 INTRODUCTION TO FROND END

The software requirement specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

USER INTERFACE

* Swing - Swing is a set of classes that provides more powerful and flexible components that are possible with AWT. In addition to the familiar components, such as button checkboxes and labels, swing supplies several exciting additions, including tabbed panes, scroll panes, trees and tables.

* Applet - Applet is a dynamic and interactive program that can run inside a web page displayed by a java capable browser such as hot java or Netscape.

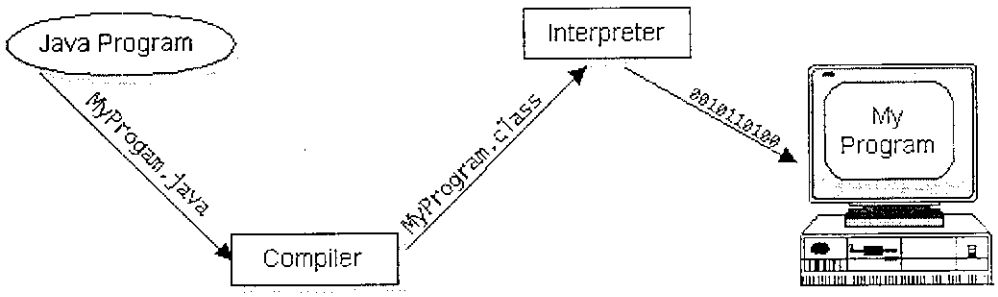
JAVA:

Java was conceived by James Gosling, Patrick Naughton , Chris Wrath, Ed Frank, and Mike Sheridan at Sun Micro system. It is an platform independent programming language that extends it's features wide over the network.Java2 version introduces an new component called "Swing" – is a set of classes that provides more power & flexible components than are possible with AWT.

- It's a light weight package, as they are not implemented by platform-specific code.
- Related classes are contained in javax.swing and its sub packages, such as javax.swing.tree.
- Components explained in the Swing have more capabilities than those of AWT.

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java byte codes--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java byte

code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how this works.



Java byte codes can be considered 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.

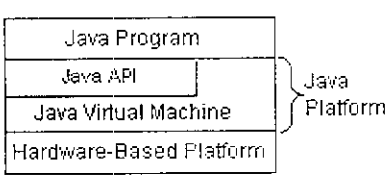
Java byte codes help make "write once, run anywhere" possible. The Java program can be compiled into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components.

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

With packages of software components that provide a wide range of functionality. The core API is the API included in every full implementation of the Java platform. The core API gives you the following features:

- The Essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- Applets: The set of conventions used by Java applets.
- Networking: URLs, TCP and UDP sockets, and IP addresses.
- Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- Security: Both low-level and high-level, including electronic signatures, public/private key management, access control, and certificates.
- Software components: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.
- Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- Java Database Connectivity (JDBC): Provides uniform access to a wide range of relational databases.
- Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.

NETWORKING BASICS

Ken Thompson and Dennis Ritchie developed UNIX in concert with the C language at Bell Telephone Laboratories, Murray Hill, New Jersey, in 1969. In 1978, Bill Joy was leading a project at Cal Berkeley to add many new features to UNIX, such as virtual memory and full-screen display capabilities. By early 1984, just as Bill was leaving to found Sun Microsystems, he shipped 4.2BSD, commonly known as Berkeley UNIX.4.2BSD came with a fast file system, reliable signals, interprocess communication, and, most important, networking. The networking support first found in 4.2 eventually became the de facto standard for the Internet. Berkeley's implementation of TCP/IP remains the primary

standard for communications with the Internet. The socket paradigm for inter process and network communication has also been widely adopted outside of Berkeley.

SOCKET OVERVIEW

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can "plug in" to the socket and communicate.

Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

CLIENT/SERVER

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

In Berkeley sockets, the notion of a socket allows as single computer to serve many different clients at once, as well as serving many different types of information. This feat is managed by the introduction of a port, which is a numbered socket on a particular machine. A server process is said to "listen" to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To mange multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

RESERVED SOCKETS

Once connected, a higher-level protocol ensues, which is dependent on which port you are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

JAVA AND THE NET

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

INET ADDRESS

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. We interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

Factory Methods

The InetAddress class has no visible constructors. To create an InetAddress object, we use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are shown here.

static InetAddress getLocalHost( ) throws UnknownHostException

static InetAddress getByName(String hostName) throws UnknowsHostException

static InetAddress[ ] getAllByName(String hostName)

throws UnknownHostException

The getLocalHost( ) method simply returns the InetAddress object that represents the local host. The getByName( ) method returns an InetAddress for a host name passed to it. If these methods are unable to resolve the host name, they throw an UnknownHostException.

On the internet, it is common for a single name to be used to represent several machines. In the world of web servers, this is one way to provide some degree of scaling. The getAllByName ( ) factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can't resolve the name to at least one address. Java 2, version 1.4 also includes the factory method getByAddress( ), which takes an IP address and returns an InetAddress object. Either an IPv4 or an IPv6 address can be used.

INSTANCE METHODS

The InetAddress class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used.

Boolean equals (Object other)- Returns true if this object has the same Internet address as other.

byte[ ] getAddress( )- Returns a byte array that represents the object's Internet address in network byte order.

String getHostAddress( )- Returns a string that represents the host address associated with the InetAddress object.

String getHostName( )- Returns a string that represents the host name associated with the InetAddress object.

boolean isMulticastAddress( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.

String toString( )- Returns a string that lists the host name and the IP address for convenience.

Internet addresses are looked up in a series of hierarchically cached servers. That means that your local computer might know a particular name-to-IP-address mapping autocratically, such as for itself and nearby servers. For other names, it may ask a local DNS server for IP address information. If that server doesn't have a particular address, it can go to a remote site and ask for it. This can continue all the way up to the root server, called InterNIC (internic.net).

TCP/IP CLIENT SOCKETS

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The ServerSocket class is designed to be a "listener," which waits for clients to connect before doing anything. The Socket class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

Socket(String hostName, int port) Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.

Socket(InetAddress ipAddress, int port) Creates a socket using a preexisting InetAddress object and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

InetAddress getInetAddress( )- Returns the InetAddress associated with the Socket object.

int getPort( ) Returns the remote port to which this Socket object is connected.

int getLocalPort( ) Returns the local port to which this Socket object is connected.

Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IOException if the sockets have been invalidated by a loss of connection on the Net.

InputStream getInputStream( )Returns the InputStream associated with the invoking socket.

OutputStream getOutputStream( ) Returns the OutputStream associated with the invoking socket.

TCP/IP SERVER SOCKETS

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets.

When we create a ServerSocket, it will register itself with the system as having an interest in client connections. The constructors for ServerSocket reflect the port number that we wish to accept connection on and, optionally, how long we want the queue for said port to be. The queue length tells the system how many client connection it can leave pending before it should simply refuse connections. The default is 50. The constructors might throw an IOException under adverse conditions. Here are the constructors:

ServerSocket(int port) Creates server socket on the specified port with a queue length of 50.

Serversocket(int port, int maxQueue)-Creates a server socket on the specified port with a maximum queue length of maxQueue.

ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multihomed host, localAddress specifies the IP address to which this socket binds.

ServerSocket has a method called accept( ), which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

6.2 INTRODUCTION TO BACK END

MICROSOFT SQL SERVER 2000

Micro soft SQL Server 2000 provides a scalable database that combines ease of use with complex analysis and data warehousing tools. SQL Server includes a rich graphical user interface (UI) along with a complete development environment for creating data- driven applications.

Commerce server takes advantages of SQL Server data warehousing and analysis capabilities in several key areas. The commerce server data warehouse, for example, uses SQL Server Data Transformation Services (DTS) to transform data stored in SQL Server database to the format used by Commerce server resources.

WINDOWS 2000 AND SQL SERVER SECURITY:

Existing Microsoft ® 2000 accounts (Active Directory users or groups) must be granted permissions to connect to Microsoft ® Microsoft ® SQL Server ™ before they can access a database. If

all members of a windows group require connections to SQL Server, you can grant login permissions for the group as a whole.

Managing group permissions is easier than managing permissions for individual users. If you do not want a group to be granted permissions collectively, you can grant permissions to connect to SQL Server for each individual user.

ACTIVE DIRECTORY USERS AND GROUPS:

In windows 2000, users are individuals who have an account that provides specific privileges to access information and resources. Granting permission to users to develop , manage , and use workflow applications is dependent upon the integration of windows 2000 domain accounts and SQL Server roles. If a number of users all have the same permissions, they can be treated as a single unit, called a group, which can be assigned permissions that apply to all members of the group. Individuals can be added to or removed from groups as desired.

There are two types of windows groups: **Global** and **Local**

Global groups contain user accounts from the windows 2000 server domain in which they are created on a computer running windows 2000 professional.

Local groups can contain user accounts and global from the domain in which they are created and any trusted domain. Local groups cannot contain other local groups. In addition, Windows 2000 has predefined, built – in local groups, such as administrators, Users and Guests. By default, these built – in groups always are available on any windows 2000 computers, unless they are removed explicitly.

To grant access to SQL Server to a windows local or global group, specify the domain or computer name on which the group is defined, flowered by a backslash, and then the group name. For example, to grant access to the windows 2000 group SQL_users in the

Windows 2000 domain LONDON, specify LONDON/SQL_users as the group name.

However, to grant access to a window built – in local group, specify BUILT IN, instead of the domain or computer name. To grant access to the built – in windows local group Administrators, specify BUILTIN/Administrator as the group name to add to SQL server.

You must have appropriate permissions on the server to create windows groups or users or to create SQL servers users or roles.For additional information about windows accounts, see your windows documentation.

SQL SERVER LOGINS

SQL Server logins are the account identifiers that control access to any SQL server will not complete connections unless it has first verified that the login you specified is valid. This verification of the login is called authentication.

A member of the SQL server sysadmin fixed- server role first must specify to SQL Server all the windows accounts or groups that can connect to SQL Server. Your access to SQL Server is controlled

by your windows account or group, which is authenticated when you log on to the windows operating system on the client.

When Connecting, the SQL Server client software requests a windows trusted connection to SQL Server. Windows will not open a trusted connection unless the client has logged on successfully using a valid windows account. The properties of a trusted connection include the windows group and user accounts of the client that opened the connection. SQL Server gets the user account information from the trusted connection properties and matches them against the windows accounts defined as valid SQL Server logins. If SQL Server finds a match, it accepts the connection. You are identified in SQL Server finds a match, it accepts the connection. You are identified in SQL Server by your windows group or user account.

DATABASE ROLES

Using database roles, you can collect users into a single unit to which you can apply permissions. Permissions granted to, denied to or revoked from a role also apply to any members of the role.

SQL Server roles exist within a database and cannot span more than one database. Because roles are unique to each database, you can reuse a role name, such as "reviewer" in each database that you create. To assign users and groups to data base roles, the users and groups must have valid windows domain accounts and SQL Server logins.

If you make any changes to the membership of database roles in your workflow application, you must synchronize the user directory for role permissions to work properly.

- Users can belong to more than one database role at a time.
- Roles can contain windows accounts and other SQL Server users and roles.
- A scalable model is provided for setting up the right level of security within a database

It is easy to manage permissions in a database if you define a set of roles based on job functions and assign each role the permissions that apply to that job. Then, you can move users between roles rather than having the permissions for each individual user.

The owner of a role determines who can be added or removed from the role. The owner is either the user explicitly specified as the owner when the role is created or the user who created the role when no owner is specified. If you make any changes to the membership of database roles in your workflow application, you must synchronize the user directory for role permissions to work properly. Database roles are created for a particular database. In SQL Server 7.0 and SQL Server 2000, users can belong to multiple roles. Because users can belong to more than one database role at a time, it is no longer required for users to assume temporarily the identify (and permissions) of other users through aliases.

Note if you plan to make a template based on a workflow application, you should use role based permissions for everything, because the set of database users will be different for each instance of a project based on the template.

DATABASE USER ACCOUNT

While a SQL Server login makes it possible for a user to access SQL Server, a database user account is required for the user to access a specific database. Then, these user accounts can be associated with the roles defined in you workflow application.

A user account can be a member of any number of roles within the same workflow application. For example, a user can be a member of the admin role and the author's role for the same database, with each role granting different permissions.

The effective permissions on an object granted to a member of more than one role are the cumulative permissions of the roles, although denied permissions in one role has precedence over the same permissions granted in another role. For example, the admin role might grant access to a table while the author's role denies access to the same table. A member of both roles is denied access to the table, because denied access is the most restrictive.

CHAPTER 7
TESTING

7.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.3 FUNCTIONAL TEST

Functional tests provide a systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation , and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify

Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Here all the modules are checked before integrating the system.Here we have shown a chart between the old TCP and New TCP. Which collects the results after transmitting the packets from existing and proposed system?

7.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

CHAPTER8

CODING

network.java

```
 import java.sql.*;
import java.io.*;
import java.net.*;
import java.util.*;

public class network
{

        private database db;
        ResultSet rs;
        Connection cs;
        Statement st;
        Vector v,v1;
        static ServerSocket sersoc;
        static Socket soc;
        ObjectInputStream dis;
        ObjectOutputStream dos;
        InputStream is;
        OutputStream os;
  String n="",ip="",s,s1,name;
  static network m;
  int port=4444,na;
        public network()throws Exception
        {
                super();
                db = new database();
                st=db.dbcon();
                 s="delete  from peerconnection";
                 s1="delete from RoutingTable";
                 db.delete(s);
                 db.delete(s1);
         }

        public void rec()throws Exception
         {


          dis=new ObjectInputStream(soc.getInputStream());
          String nname=(String)dis.readObject();
          System.out.println("recived:"+nname);

          if(nname.equals("peers"))
          {
                v=new Vector();
                s="select * from peerconnection";
                v=db.select(s);
                System.out.println("v:"+v);
                dos=new ObjectOutputStream(soc.getOutputStream());
```

```
                dos.writeObject(v);

        }
        else if(nname.equals("exit"))
        {
                dis=new ObjectInputStream(soc.getInputStream());
                String nname1=(String)dis.readObject();
                System.out.println("exit:"+nname1);
                 v=new Vector();
                s="delete from peerconnection where peerno='"+nname1+"'";
                db.delete(s);
        }
        else if(nname.equals("user"))
        {
                dis=new ObjectInputStream(soc.getInputStream());
                String nname2=(String)dis.readObject();
                String array[]=nname2.split("&");
                System.out.println("nname2:"+nname2);
                System.out.println("array[0]:"+array[0]);
                System.out.println("array[1]:"+array[1]);
                int q=Integer.parseInt(array[1]);
                System.out.println("q:"+q);
                s="select * from peerconnection where peerno='"+array[0]+"'";
                if(db.check(s))
                {
                        s="update peerconnection set status='true' where peerno='"+array[0]+"'";
                        db.insert(s);
                 }
                 else
                 {

                        s="update peerconnection set  peerno='"+array[0]+"'where port='"+q+"'";
                        System.out.println("q:"+q);
                        db.insert(s);
                        System.out.println("q:"+q);
                }
         }
         else if(nname.equals("Routing Table"))
         {
                        //dis=new ObjectInputStream(soc.getInputStream());
                        String FileName=(String)dis.readObject();
                        String SystemName=(String)dis.readObject();
                        String NodeName=(String)dis.readObject();

                  if(!FileName.equals(""))
                  {
                                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                                Connection con=DriverManager.getConnection("jdbc:odbc:net");
                                Statement st=con.createStatement();
                                ResultSet rs=st.executeQuery("Select * from RoutingTable where
FileName='"+FileName+"'");
                                if(rs.next())
                                {
                                        //System.out.println("FileName already in DataBase");
```

```
                                }
                                else
                                {
                                        String s1="insert into RoutingTable
values('"+NodeName+"','"+FileName+"','"+SystemName+"')";
                                        db.insert(s1);
                                        System.out.println("FileName   :::::: "+FileName);
                                        System.out.println("SystemName :::::: "+SystemName);
                                        System.out.println("NodeName   :::::: "+NodeName);
                                }
                        }


                }

}
```

peer.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import org.gui.JDirectoryDialog;
public class peer extends JFrame implements Runnable
{
   public static Socket sousoc1;//,ss1;
   ObjectInputStream oi,oi1;
   ObjectOutputStream oo,oo1;
   static Thread th;
```

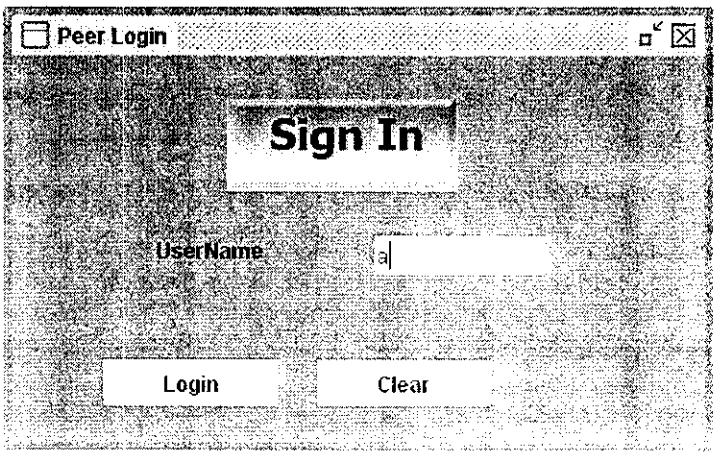
```
   int t,p,k,i;
   String data1=new String();
   String rec1="",port;
   static String name;
   ResultSet rs,rs1;
   Connection c;
   Statement s,s1;
   Vector v1;
   Vector f,f1;
   String ia="";
   byte bd[];
   String filelist[];
   String peerdetails[][];
   public static peerUI pf1;
   public JDirectoryDialog directoryDialog;
String desdir="",network;
save dp;
File f2,f5;
String username,u1;
login u;
sysname sys=new sysname();

Vector fil=new Vector();

  public peer()throws Exception
        {
          dbaseconnect1();
```

```
        s.executeUpdate("delete from RoutingTable1");
        network=sys.orgin();
         th=new Thread(this);
      th.start();

      }
public void dbaseconnect1()throws Exception
      {

            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            c=DriverManager.getConnection("jdbc:odbc:searchpeer1");
            s=c.createStatement();
    s1=c.createStatement();
      }

public void ownfiles1(String username)throws Exception
      {
            File fi=new File(".","//peersharing");
            filelist=fi.list();
            v1=new Vector();
            for(int i=0;i<filelist.length;i++)
            {
                  s.executeUpdate("Insert into RoutingTable1
values('"+username+"','"+filelist[i]+"','0')");
            }
  }
```

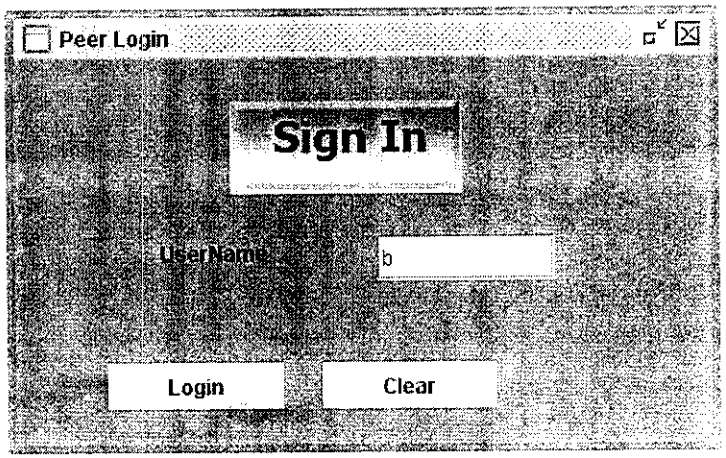
CHAPTER 9

SCREEN SHOTS

Peer login form

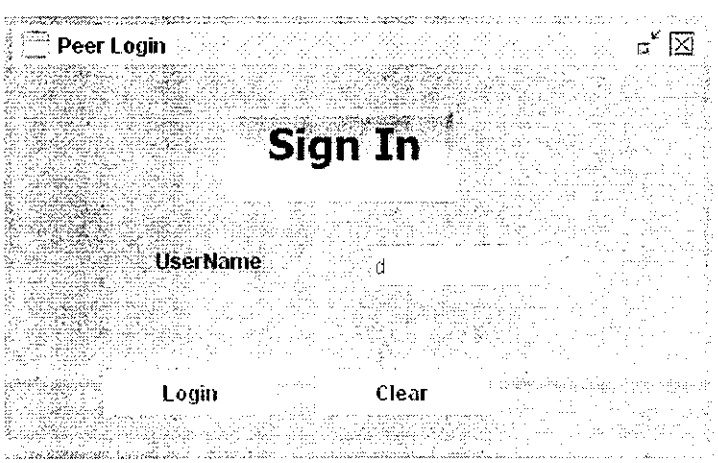


Peer home page

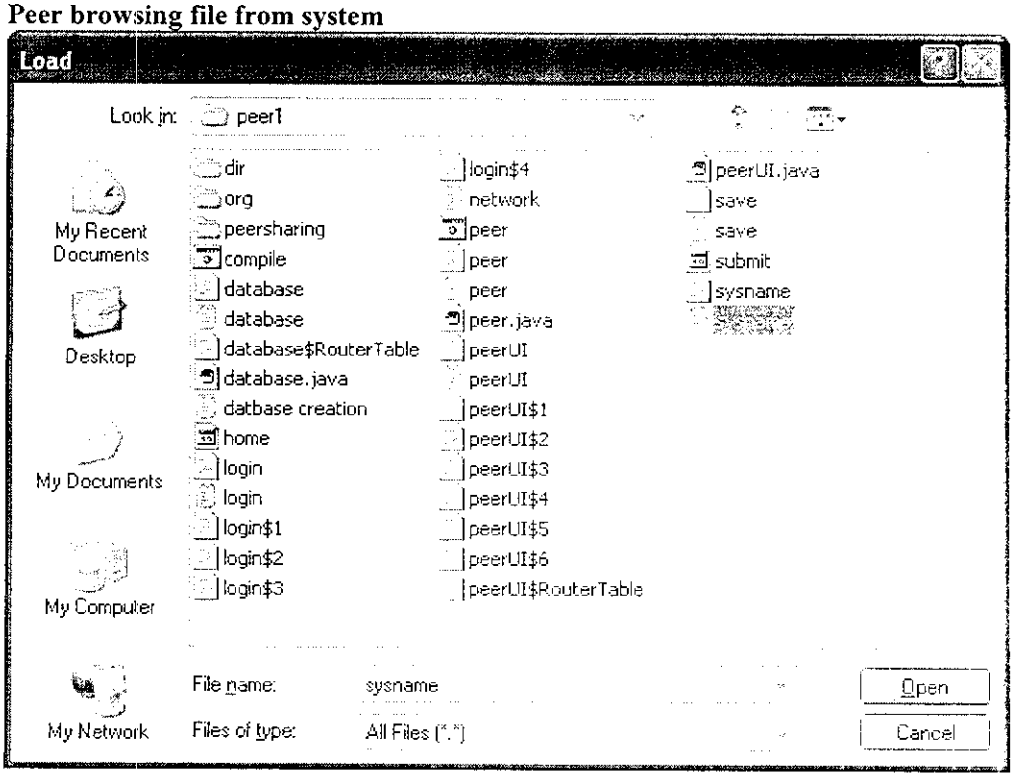

Peer Login
Sign In
UserName
b
Login
Clear


b
Welcome
Frame
File
Peer
RouTable
Choose File
Browse
Search
Share
Leave

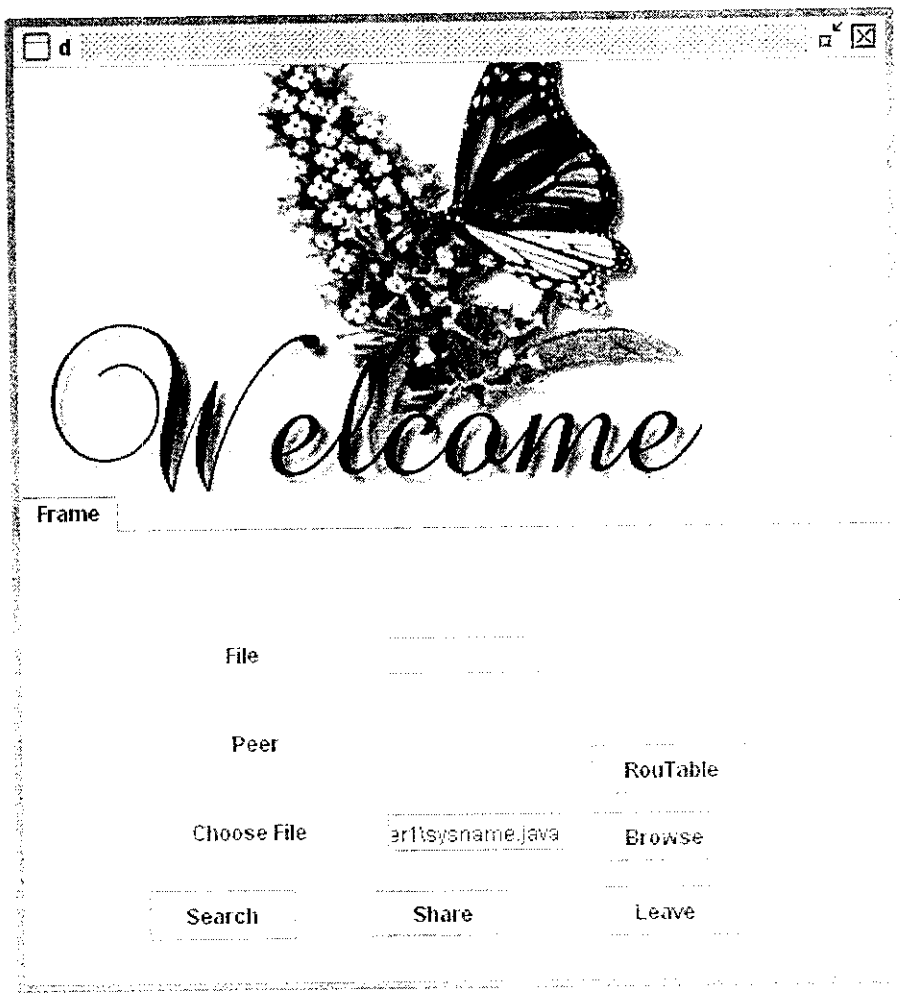

c
Welcome
Frame
File
Peer
RouTable
Choose File
Browse
Search
Share
Leave


Peer Login
Sign In
UserName
d
Login
Clear

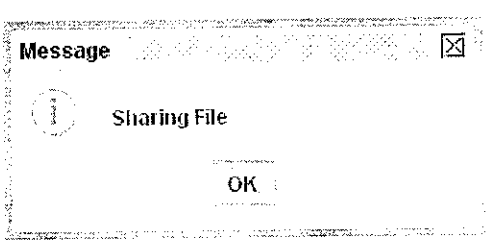
Peer browsing file from system



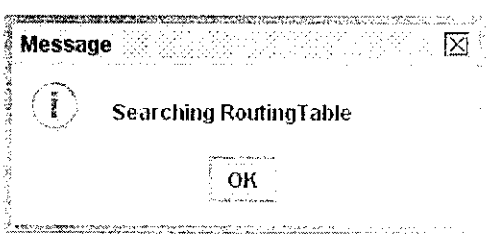
Peer is sharing the file



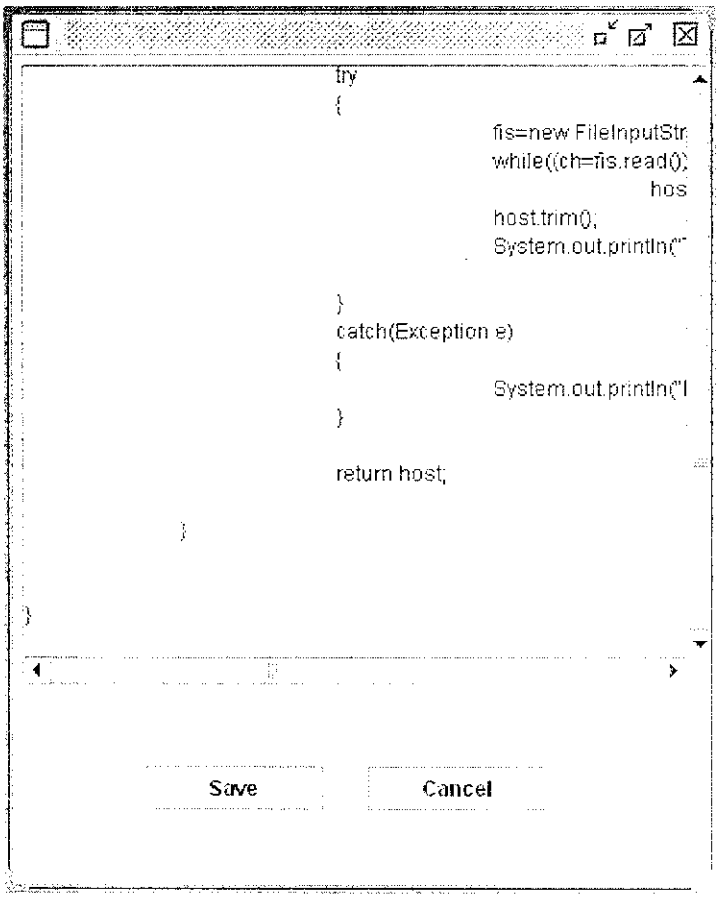
File has been shared



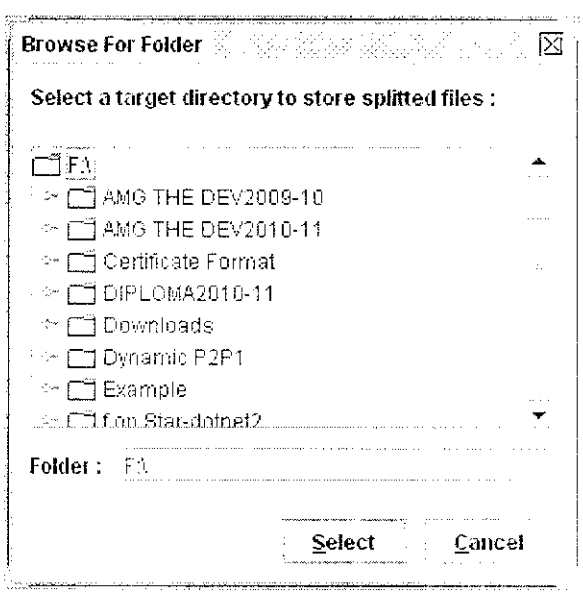
Searching file from routing table



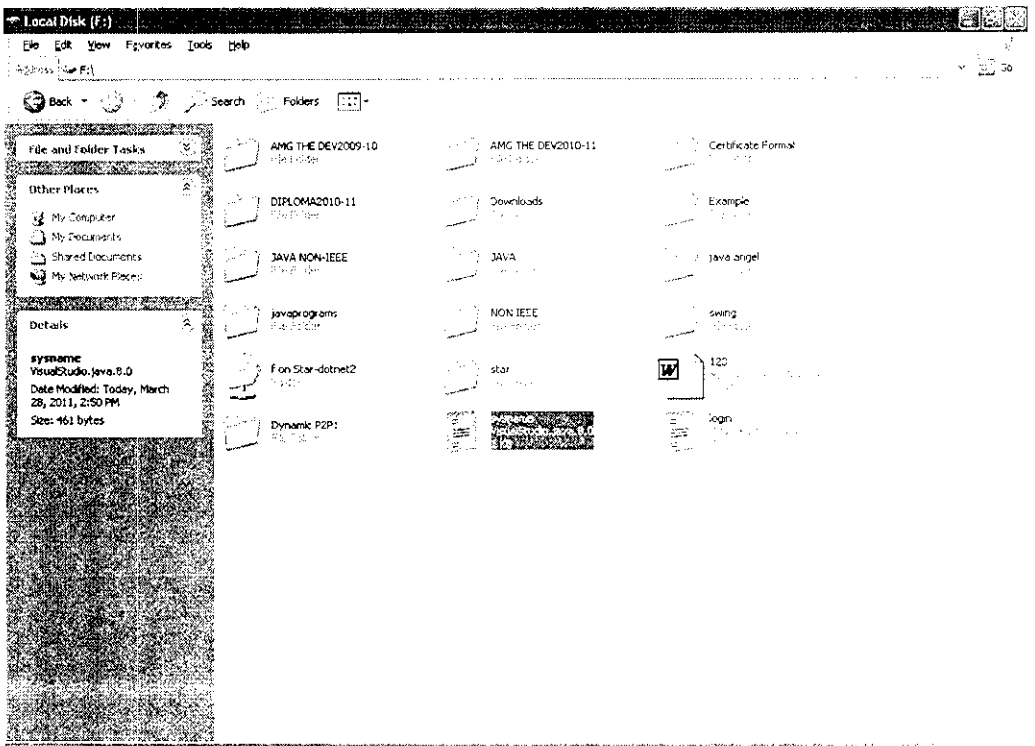
File has found



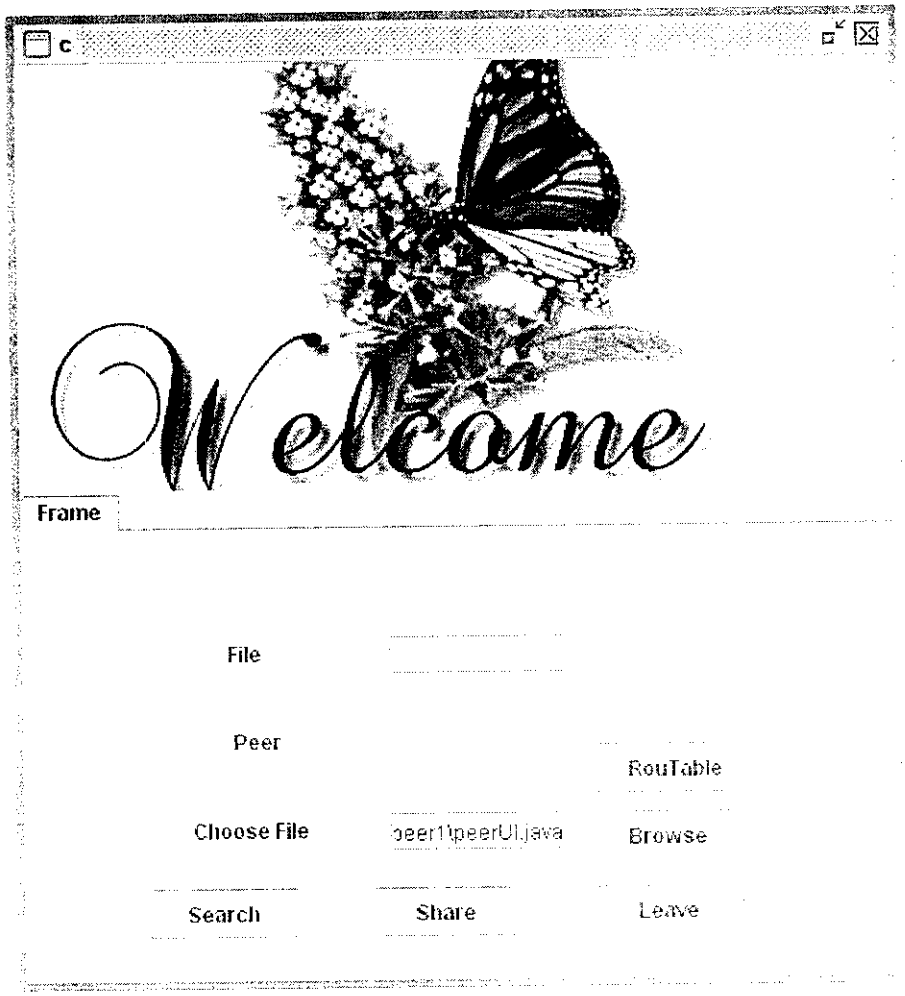
File has been displayed and saved

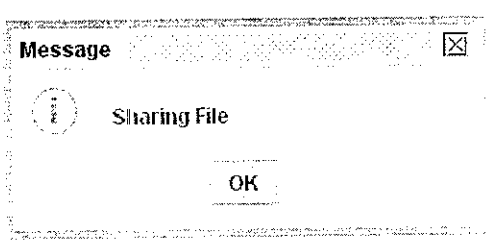


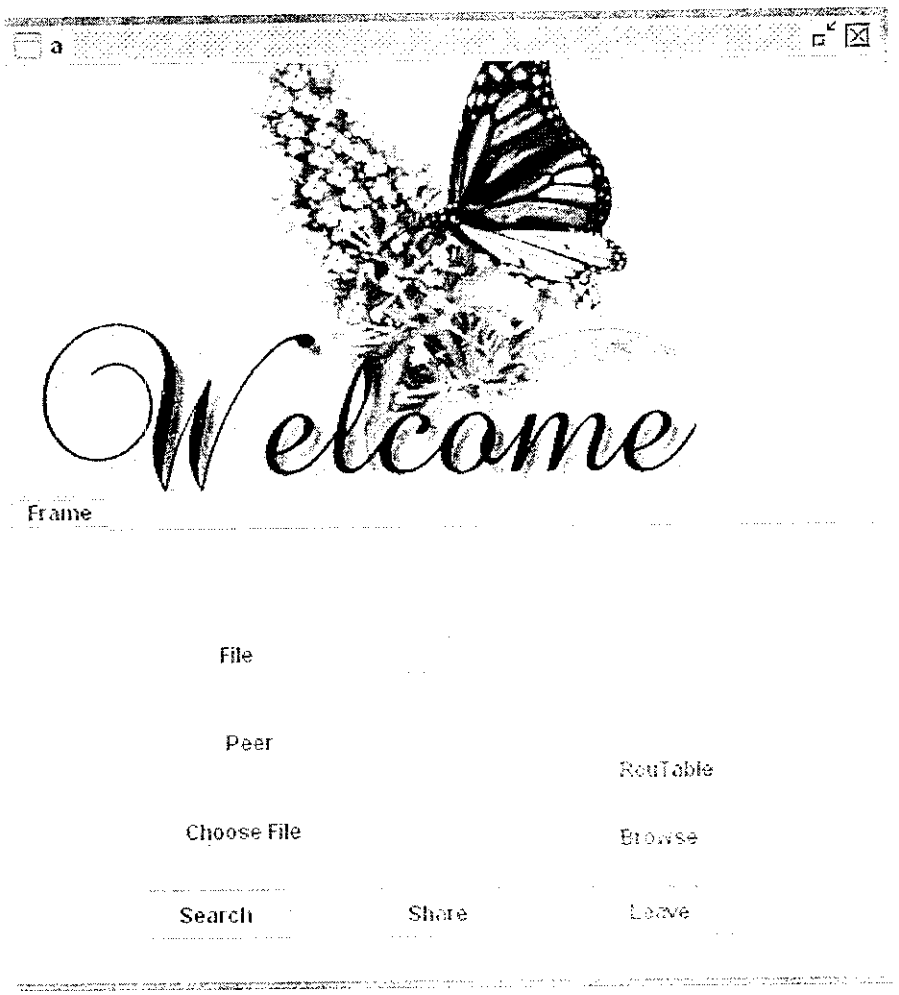
Selecting the folder for saving the file



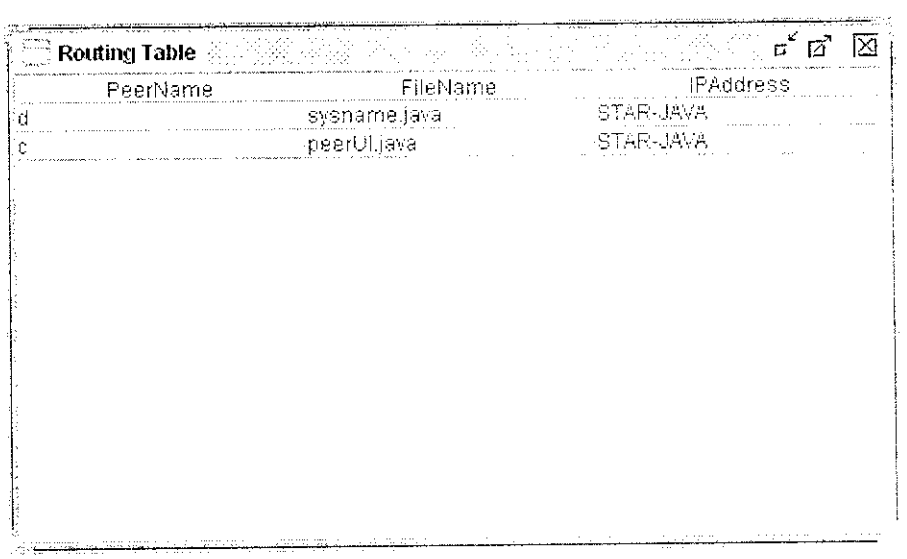
File displaying in system after saving


c
Welcome
Frame
File
Peer
RouTable
Choose File
peer1\peerUI.java
Browse
Search
Share
Leave


Message
Sharing File
OK



Displaying routing table



CHAPTER 11

REFERENCES

1. D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger "Sampling Techniques for Large, Dynamic Graphs," Proc. Ninth IEEE Global Internet Symp. (Global Internet '06), Apr. 2006

2. A.H. Rasti, D. Stutzbach, and R. Rejaie, "On the Long-Term Evolution of the Two-Tier Gnutella Overlay," Proc. Ninth IEEE Global Internet Symp. (Global Internet '06), Apr. 2006.

3. D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," Technical Report HPL-2002-57, HP, 2002.

4. K. Sripanidkulchai, The Popularity of Gnutella Queries and Its Implications on Scalability, white paper, Carnegie Mellon Univ., Feb. 2001.
5. M. Jovanovic, F. Annexstein, and K. Berman, "Scalability Issues in Large Peer-to-Peer Networks: A Case Study of Gnutella," technical report, Laboratory for Networks and Applied Graph Theory, Univ. of Cincinnati, 2001.

6. B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02), pp. 5-14, July 2002.
7. www.wikipedia.org