Chapter 1

INTRODUCTION

1.1 Overview

Today, we are more and more dependent on Internet services, which provide important functionality and store critical state. These services are often implemented on collections of machines residing at multiple geographic locations such as a set of corporate data centres. For example, Dynamo uses tens of thousands of servers located in many data centres around the world to build a storage back-end for Amazon's S3 storage service and its e-commerce platform. As another example, in Google's cluster environment each cluster includes an installation of the GFS file system spanning thousands of machines to provide a storage substrate. Additionally, these systems are long lived and need to continue to function even though the machines they run on break or are decommissioned. Thus, there is a need to replace failed nodes with new machines; also it is necessary to add machines to the system for increased storage or throughput. Thus, the systems need to be reconfigured regularly so that they can continue to function.

Our project implements a paper which provides a complete solution for reliable, automatic reconfiguration in distributed systems.

1.2 Objectives

The objective of this project is to create a system that provides the following features:

- It must provide an abstraction of a globally consistent view of the system membership. This abstraction simplifies the design of applications that use it, since it allows different nodes to agree on which servers are responsible for which subset of the service.
- It must be designed to work at large scale, e.g., tens or hundreds of thousands of servers. Support for large scale is essential since systems today are already large and we can expect them to scale further.
- It should be secure against Byzantine (arbitrary) faults. Handling Byzantine faults is important because it captures the kinds of complex failure modes that have been reported

or our target deployments. Additionally, the Byzantine fault model makes it possible to tolerate malicious intrusions where an attacker gains control over a number of servers.

1.3 Summary

The motive of the project is to provide a reliable and automatic means to check the intrusions of attackers, or users with malicious intent, from harming the system. Critical systems that provide important functions to millions of people are always susceptible to the few aberrations in their user-kind. This means that there will always be a few users who try to tamper with the system and try to access some services which have been denied to them explicitly. Since the system is providing very critical services, it is essential that the main system be accessible to regular and honest users without any delay. Therefore, the main system's working should not be paused or halted even for an instant, in the process of monitoring the system for intrusions.

Byzantine fault tolerant systems provide this functionality that is they disable attackers from accessing the system but continue to provide uninterrupted access to their legitimate users. We are particularly interested in applying the notion of Byzantine tolerance to large-scale storage systems which are critical systems containing large amounts of data, providing services to several users. The project is aimed at ensuring that the information on these systems is not compromised, and at the same time service to legitimate users is not disrupted.

Chapter 2

LITERATURE SURVEY

2.1 Introduction

Literature survey is the most important step in software development process. We referred to the papers mentioned below.

2.1.1 Authentication in a Reconfigurable Byzantine Fault Tolerant System:

Byzantine faults occur as a result of software errors and malicious attacks; they are increasingly a problem as people have come to depend on online services. Systems that provide critical services must behave correctly in the face of Byzantine faults. Correct service in the presence of failures is achieved through replication: the service runs at a number of replica servers and as more than a third of the replicas are non-faulty, the group as a whole continues to behave correctly. We would like the service to be able to authenticate data. Authenticated data is data that more than a third of the service is willing to sign.

If a long-lived replicated service can tolerate ‘f’ failures, then we do not want the adversary to have the lifetime of the system to compromise more than ‘f’ replicas. One way to limit the amount of time an adversary has to compromise more than ‘f’ replicas is to reconfigure the system, moving the responsibility for the service from one group of servers to a new group of servers. Reconfiguration allows faulty servers to be removed from service and replaced with newly introduced correct servers. Reconfiguration is also desirable because the servers can become targets for malicious attacks, and moving the service thwarts such attacks.

2.1.2 Secure routing for structured peer-to-peer overlay networks

Structured peer-to-peer overlay networks provide a substrate for the construction of large-scale, decentralized applications, including distributed storage, group communication, and content distribution. These overlays are highly resilient; they can route messages correctly even when a large fraction of the nodes crash or the network partitions. But current overlays are not secured; even a small fraction of malicious nodes can prevent correct message delivery throughout the overlay. This problem is particularly serious in open peer-to-peer systems, where many diverse, autonomous parties without preexisting trust relationships wish to pool their

resources. This paper studies attacks aimed at preventing correct message delivery in structured peer-to-peer overlays and presents defenses to these attacks.

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture

All computer operating systems are designed for particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory

All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of

other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage

Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter

Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals

Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

2.3Software Requirements

This section specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) includes overview as well as the functional and non-functional requirement of this dissertation.

A *Software Requirements Specification* (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. SRS also contains non-functional (or supplementary) requirements.

Software Requirement Specification (SRS):

Functional	Control of Misbehaving Users at Gateway; Fraud Detections.
Non- Functional	Server never controls the Manager.

External interface	WiFi LAN , GateWay
Performance	Detecting Frauds based on the membership.
Attributes	Membership Deposit, Membership Revocation, membership Issuance, Fraud Detection, Misbehaving Users.

Table: 3.1 Summary of SRS

A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish. Non-functional requirements are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. They apply to the system as a whole. Functional requirements are supported by non-functional requirements.

2.3.1 Functional Requirements

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organization when writing requirements. Functional system requirements describe the system function in detail, its inputs and outputs, exceptions, and so on.

In principle, the functional requirements specification of a system should be both complete and consistent. Completeness means that all services required by the user should be defined. Consistency means that the requirements should not have contradictory definitions.

A few examples of functional requirements for our system are:

1. Every remote client will be allocated a unique secret key, which will allow user to access the services she signed up for.

2. The owner will be able to view the list of fraudulent users and remove/shut down specific users.

3. The list of fraudulent users is automatically updated in the database whenever some unauthorized access takes place using wrong key.

2.3.2 Non – Functional Requirements

Performance

The performance of the developed applications can be calculated by using following methods. Measuring enables you to identify how the performance of your application stands in relation to your defined performance goals and helps you to identify the bottlenecks that affect your application performance. It helps you identify whether your application is moving toward or away from your performance goals. Defining what you will measure, that is, your metrics, and defining the objectives for each metric is a critical part of your testing plan.

Performance objectives include the following:

- Response time or latency
- Throughput
- Resource utilization

Safety and Security

The security design process is cyclic. The security of an application depends upon the vigilance of the developers and administrators not just during the design phase but also for the life of the application. Since new threats arise almost daily, an application must be scrutinized constantly for potential security flaws. However, the initial design of an application determines how often those flaws are likely to occur.

Security threats are any potential occurrence, malicious or otherwise, that can have an undesirable effect on the application. Vulnerabilities in the application or operating system make a threat possible. An attack on the application is an action taken by a malicious intruder to exploit certain vulnerabilities in order to enact the threat. The risk involved is the potential damage that attack can inflict on the application or even the business.

Maintainability

The increased complexity of modern software applications also increases the difficulty of making the code reliable and maintainable. In recent years, many software measures, known as code metrics, have been developed that can help developers understand where their code needs rework or increased testing. Developers can use Visual Studio Application Lifecycle Management to generate code metrics data that measure the complexity and maintainability of their managed code. Code metrics data can be generated for an entire solution or a single project.

Portability

Portability is one of the key concepts of high-level programming. Portability is the software code base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another. The pre-requirement for portability is the generalized abstraction between the application logic and system interfaces. When one is targeting several platforms with the same application, portability is the key issue for development cost reduction.

2.4 Summary

Software Specification

Operating System : Windows Family.

Language : Java

Front End : JAVA, Swing (JFC), J2ME

Tool : Eclipse, j2me wireless toolkit 2.5.2

Hardware specification

Processor : Pentium 4 and above

Ram : 512Mb and above.

Hard Disk : 40 Gb.

Input device : Standard Keyboard and Mouse.

Output device : VGA and High Resolution Monitor.

Network device : Ethernet LAN adapter

Chapter 3

SCOPE FOR PRESENT WORK

3.1 Statement of Project

The project implements two mainfunctions:

1. MEMBERSHIP SERVICE (MS)

   - *Tracks and responds to membership changes*

     The MS responds to requests to add and remove servers. We envision a managed environment with admission control since; otherwise, the system would be vulnerable to a Sybil attack where an adversary floods the system with malicious servers.

   - *Provides globally consistent view of the system membership*

     Periodically, the MS publishes a new system membership; in this way it provides a globally consistent view of the set of available servers. The choice of strong consistency makes it easier to implement applications, since it allows clients and servers to make consistent local decisions about which servers are currently responsible for which parts of the service.

2. AUTOMATIC RECONFIGURATIONS

   - *Reduces manual configuration errors*

     The MS works mostly automatically, and requires only minimal human intervention; this way we can reduce manual configuration errors, which are a major cause of disruption in computer systems

The first is a membership service (MS) that tracks and responds to membership changes. The MS works mostly automatically, and requires only minimal human intervention; this way we can reduce manual configuration errors, which are a major cause of disruption in computer systems periodically, the MS publishes a new system membership; in this way it provides a

globally consistent view of the set of available servers. The second part of our solution addresses the problem of how to reconfigure applications automatically as system membership changes.

3.2 Specific Objectives

The objectives of the system areas follows:

- It provides the abstraction of a globally consistent view of the system membership. This abstraction simplifies the design of applications that use it, since it allows different nodes to agree on which servers are responsible for which subset of the service.
- It is designed to work at large scale, e.g., tens or hundreds of thousands of servers. Support for large scale is essential since systems today are already large and we can expect them to scale further.
- It is secure against Byzantine (arbitrary) faults. Handling Byzantine faults is important because it captures the kinds of complex failure modes that have been reported for our target deployments. Additionally, the Byzantine fault model makes it possible to tolerate malicious intrusions where an attacker gains control over a number of servers.

Chapter 4

PRESENT WORK

4.1 Introduction

A complete solution for reliable, automatic reconfiguration in distributed systems should achieve following properties:

- **Scalable**

  Support for large scale is essential since systems today are already large and we can expect them to scale further.

- **Secure against Byzantine faults**

  Handling Byzantine faults is important because it captures the kinds of complex failure modes that have been reported for our target deployments. Additionally, the Byzantine fault model makes it possible to tolerate malicious intrusions where an attacker gains control over a number of servers.

- **Consistent**

  It provides the abstraction of a globally consistent view of the system membership. This abstraction simplifies the design of applications that use it, since it allows different nodes to agree on which servers are responsible for which subset of the service.

Earlier proposals for keeping track of a dynamic system membership do not provide all three properties. Existing Byzantine-fault tolerant systems are a static set of replicas. Many do not tolerate byzantine failures. Some that handle byzantine faults provide consistency but are not scalable and those which are scalable are not consistent.So, scalability and consistency is a problem especially for systems that are large and long-lived.

4.2Block Diagrams

4.2.1 Use Case Diagram

A **use case diagram** in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. Use case diagram for our project is shown below:

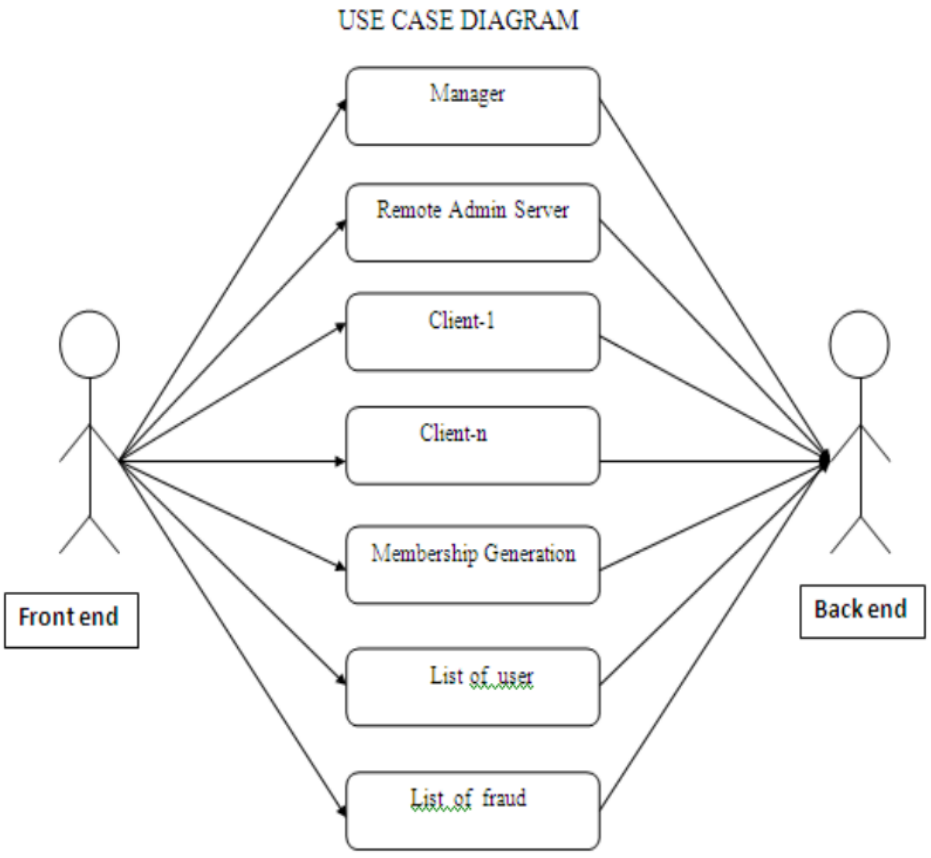

Fig. 8.1.1 Use Case

Figure 4.3.1.1 Use case diagram

4.2.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagram for our project is shown below:

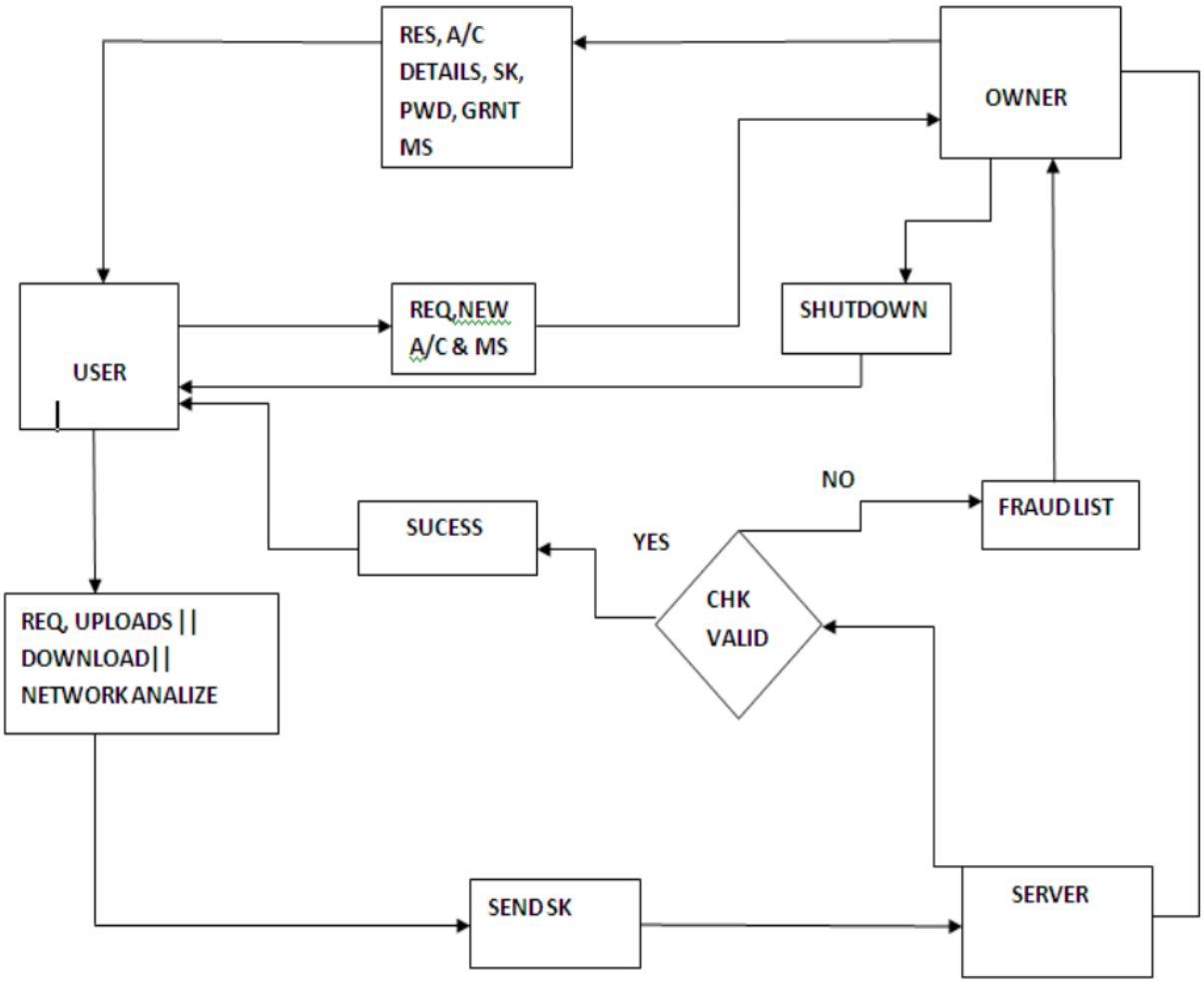

Figure 4.3.2.1 Activity diagram

4.2.3 Class Diagram

The class diagram for a manager in our project is shown below:

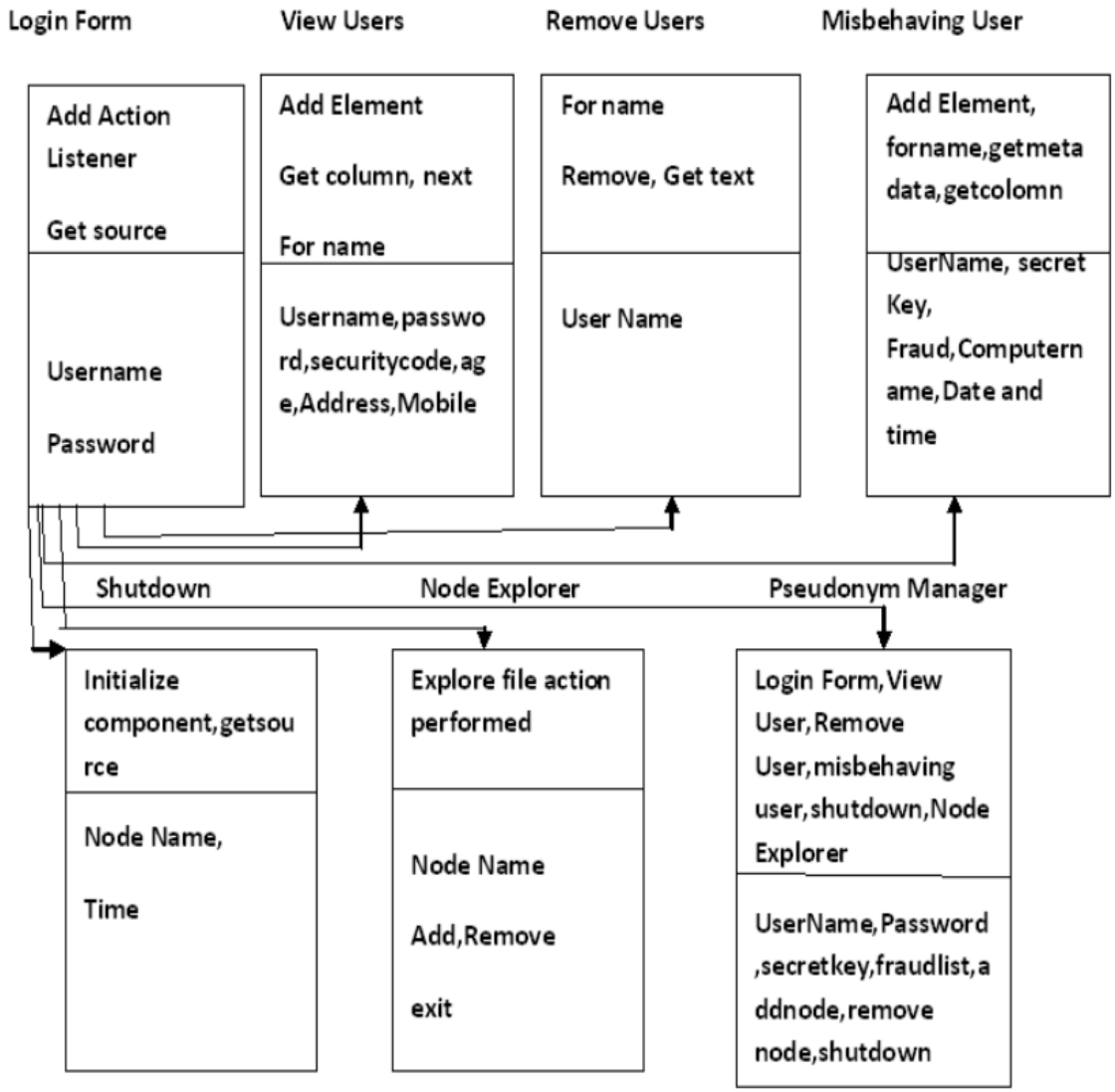

Figure 4.3.3.1 Class Diagram: Manager

The class diagram for a node in our project is shown below:

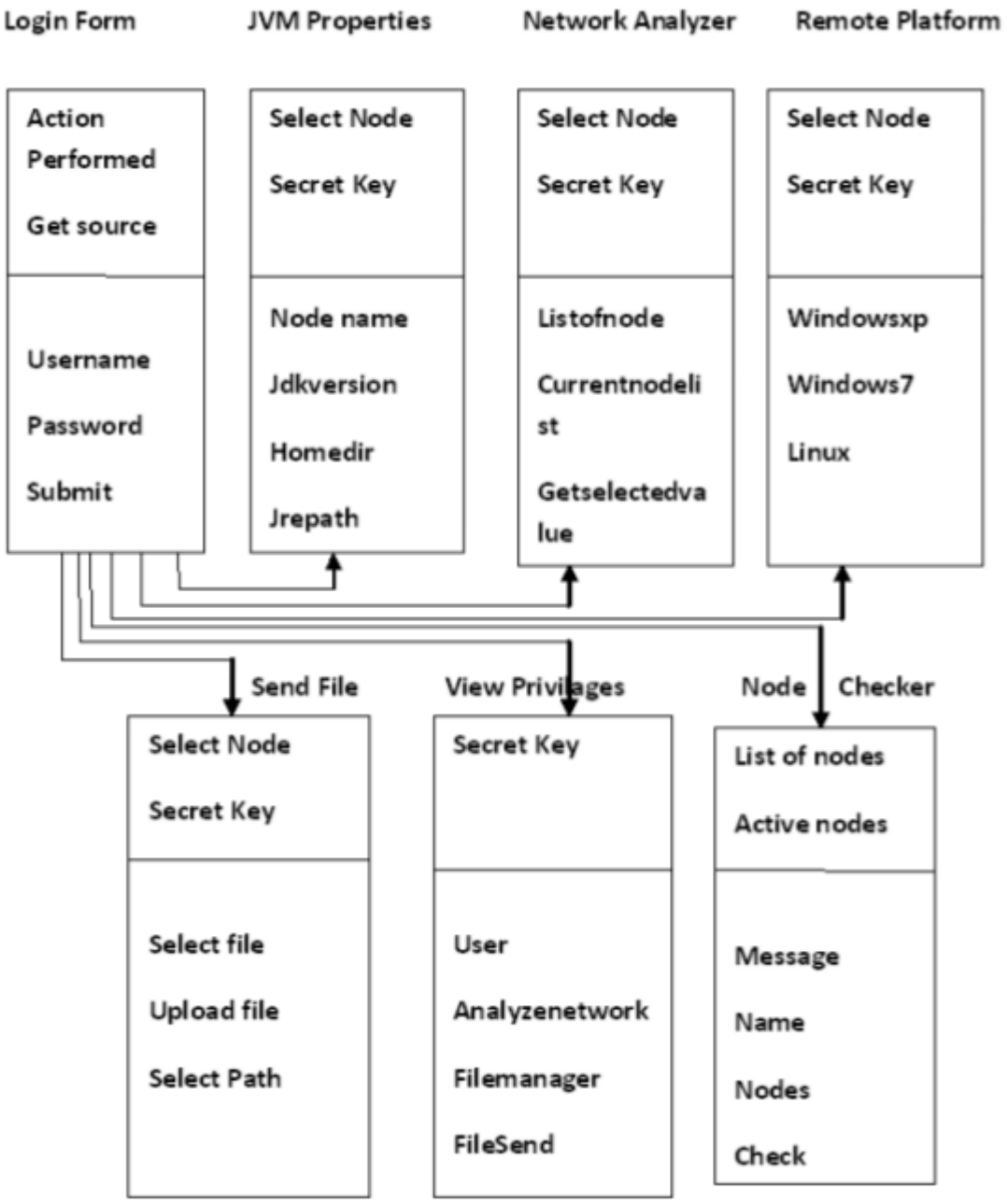

Fig. 4.3.3.2 Class Diagram: Node

4.2.4 Sequence Diagram

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. The sequence diagram for a remote client, manager and server are shown below:

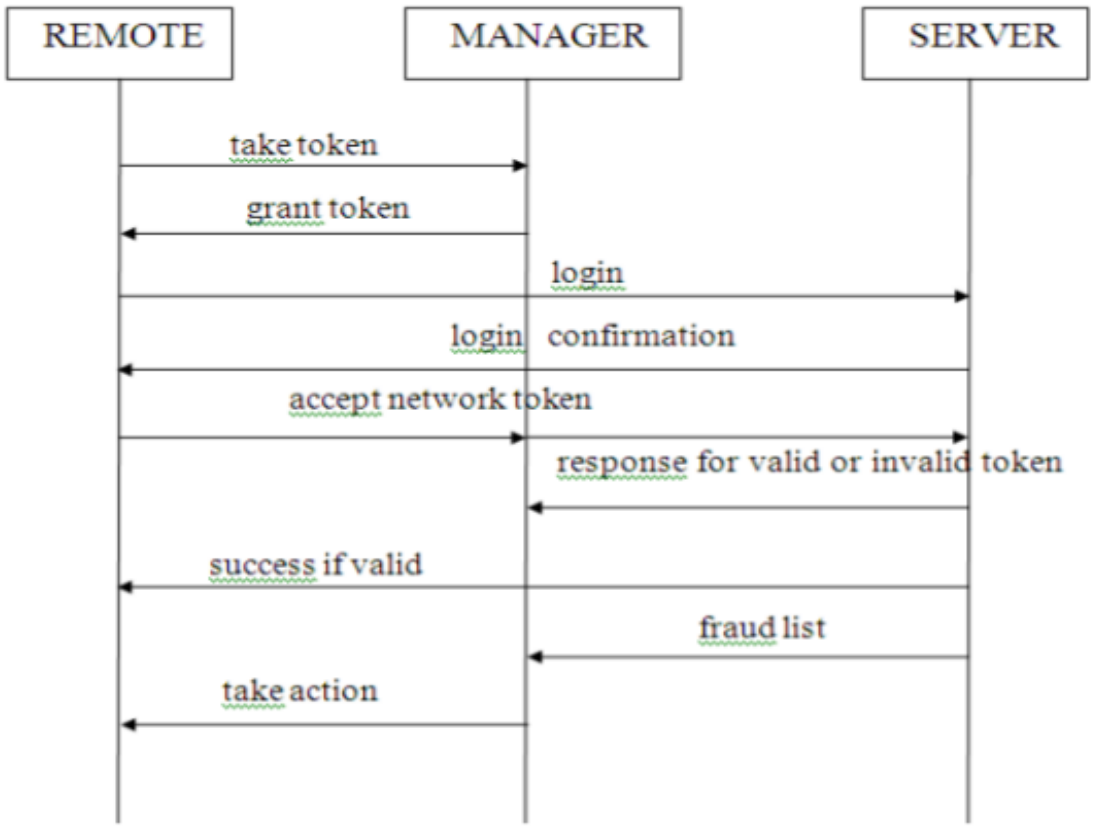

Figure 4.3.4.1 Sequence Diagram

4.2.5 Project Lifecycle

The following diagram depicts the general design and development process:

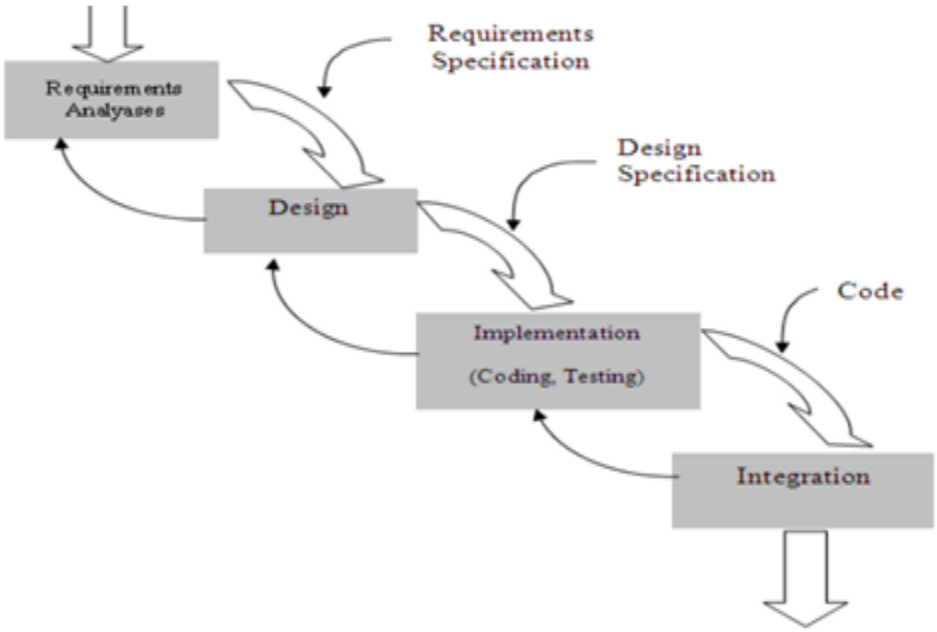

Figure 4.3.5.1 Project Development Life Cycle

4.2.6 Requirement Analysis

Requirement analysis diagram is represented below:

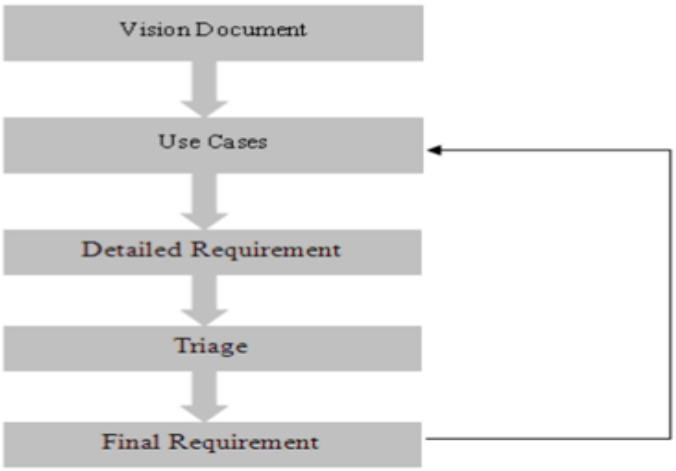

Figure 4.3.6.1 Requirements Analysis

4.3Working

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. Implementation of any software is always preceded by important decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as the real environment in which the system works the speed that is required, the security concerns, other implementation specific details etc. There are two major implementation decisions that have been made before the implementation of this project. They are as follows:

1. Selection of the programming language for development of the system

2. Selection of the platform (operating system)

4.3.1 Programming Language Selection

The Java programming language is a high-level language that can be characterized asSimple, Secure, Portable, Object oriented, Robust, Multithreading, Architecture neutral, highperformance, Distributed and Dynamic. The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives us the following features:

Development Tools

The development tools provide everything need for compiling, running, monitoring, debugging, and documenting the applications. The main tools are the javac compiler, the java launcher, and the javadoc documentation tool which are integrated in an IDE JCreater/Eclipse Galileo.

Application Programming Interface (API)

The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more. User Interface

Toolkits: The Swing and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).

Integration Libraries

Integration libraries such as the Java IDL API, JDBCTM API, Java Naming and Directory Interface TM ("J.N.D.I.") API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

Classes and functions

Java offers a rich set of classes and functions for immutable arbitrary-precision integers,java.math.BigInteger. BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

4.3.1.1 GUI Selection

Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT (abstract window toolkit). In addition to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables. Even familiar components such as buttons have more capabilities in Swing. For example, a button may have both an image and a text string associated with it. Also, the image can be changed as the state of the button changes. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and, therefore, are platform-independent. The term lightweight is used to describe such elements. The Swing-related classes are contained in javax.swing and its sub packages.

4.3.1.2 Integrated Development Environment

Eclipse is a multi-language Integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software (although it is incompatible with the GNU General Public License). It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Platform

- The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system, in contrast to some other applications, in which functionality is hard coded. The Eclipse Platform's runtime system is based on Equinox, an implementation of the OSGi core framework specification.
- This plug-in mechanism is a lightweight software componentry framework. In addition to allowing the Eclipse Platform to be extended using other programming languages such as C and Python, the plug-in framework allows the Eclipse Platform to work with typesetting languages like LaTeX, networking applications such as telnet and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with support for other version control systems provided by third-party plug-ins.

- With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal".Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.
- The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a *workspace*, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.
- Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate graphical user interface layer called JFace, which simplifies the construction of applications based on SWT.

4.4.2 Platform Selection

Even though there is no dependence of the working of the project on a particular platform, Windows XP is chosen to be the platform. However the backend of this software works in any of the operating systems, it may be the Linux platform or the windows platform with the limitation of the database used. Windows XP is popular for its new features. First and foremost is that Windows XP doesn't need any external drivers to run the hardware. It has universal support for hardware.

There is no need to install programs before using a disc or USB. Windows XP can be used for a range of computers. It exceeded some operating system like Linux, even though Linux is a freeware. This is because of the user friendly interface of Windows XP. Microsoft redesigned whole graphics user interface for Windows XP. One reason for the popularity of Windows XP is its GUI. Windows XP support wireless networking. Windows XP provides tools and programs designed to keep computer safe, manage system, and perform regularly scheduled maintenance that keeps computer running at optimum performance.

Highly appreciated feature in Windows XP is its plug and play features. The information about an active network connection like the duration of a connection and the speed at which initially connected can be obtained by using the Status menu command.

4.4 Modular Description

Our system has the following two main modules:

1. Tracking Membership Service
2. Dynamic Replication

4.4.1 Tracking membership Service

- Describes membership changes by producing a configuration.

  The MS describes membership changes by producing a configuration, which identifies the set of servers currently in the system, and sending it to all servers.

- Authenticates it using a signature that can be verified with a well-known public key

  To allow the configuration to be exchanged among nodes without possibility of forgery, the MS authenticates it using a signature that can be verified with a well-known public key.

- Functions of MS

  ✓ Membership Change Requests

  The MS responds to requests to add and remove servers. Thus, we assume servers are added by a trusted authority that signs certificates used as parameters to these requests.

  ✓ Probing

  The MS detects unreachable servers and marks them as inactive. To do this, the MS probes servers periodically, normally using unauthenticated ping messages, which we expect to be sufficient to detect most unreachable servers.

  ✓ Ending Epochs

To determine when the epoch ends, the MS tracks the termination condition. When the termination threshold is reached the MS stops probing, and produces an epoch certificate signed by the MS's private key. The signature in the certificate covers a digest of the membership (list of servers and their reachability status) and the epoch number of the new epoch.

✓ Freshness

Clients of the application using the MS need to verify the freshness of their configuration information to ensure they are communicating with the group that currently stores an item of interest, and not an old group (which may have exceeded the failure threshold). We provide freshness by means of freshness certificates.

This module is only part of what is needed for automatic reconfiguration. We assume nodes are connected by an unreliable asynchronous network like the Internet, where messages may be lost, corrupted, delayed, duplicated, or delivered out of order. While we make no synchrony assumptions for the system to meet its safety guarantees, it is necessary to make partial synchrony assumptions for live-ness. The MS describes membership changes by producing a configuration, which identifies the set of servers currently in the system, and sending it to all servers. To allow the configuration to be exchanged among nodes without possibility of forgery, the MS authenticates it using a signature that can be verified with a well-known public key.

4.4.2 Dynamic Replication

Using a single, centralized, server is the simplest way to implement a service; however, the resulting service can only be as fault tolerant as the processor executing that server. Replicas of a single server can be executed on separate processors of a distributed system, and protocols are used to coordinate client interactions with these replicas. The physical and electrical isolation of processors in a distributed system ensures that server failures are independent, as required.

- dBQS is a strongly consistent distributed hash table(DHT) that provides two types of objects. Public-key objects are mutable and can be written by multiple clients, whereas content-hash objects are immutable: once created, a content-hash object cannot change.

- The ID of a public-key object is a hash of the public key used to verify the integrity of the data.
- When a client fetches an object its integrity can be checked by verifying the signature using a public key that is also validated by the ID of the object.
- Servers are assigned random IDs in the same ID space as objects, and the replica group responsible for ID ‘i’ consists of the first ‘3f +1’ servers in the current system membership (as dictated by the MS) whose identifiers are equal to or follow i in the identifier space.
- dBQS ensures that concurrent accesses to the entire set of public-key objects are atomic: all system operations appear to execute in a sequential order that is consistent with the real-time order in which the operations actually execute.

4.5 Hardware Design

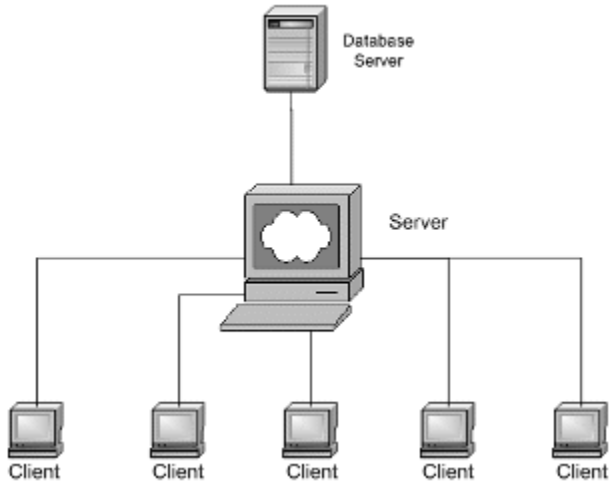

Figure 4.5.1 Hardware Design

The hardware requirements for a Local Area Network(LAN) are as follows:

1. Modem or Modem/router combo (or a connection to the internet)
2. Router (not needed if you have a modem/router combo device)
3. Network Interface Card for the devices that you want to use (or nothing if a device can pick up a wi-fi signal)

4. Internet Service Provider

5. A few Ethernet cables

Hardware specification

Processor	:	Pentium 4 and above
Ram	:	512Mb and above.
Hard Disk	:	40 Gb.
Input device	:	Standard Keyboard and Mouse.
Output device	:	VGA and High Resolution Monitor.
Network device	:	Ethernet LAN adapter

4.6 Software Design

4.6.1 Java Technology

Java technology is both a programming language and a platform.

Java Programming Language

The Java programming language **is** a high-level language that can be characterized by all of the following buzzwords:

- ✓ Simple
- ✓ Architecture neutral
- ✓ Object oriented
- ✓ Portable
- ✓ Distributed
- ✓ High performance
- ✓ Interpreted
- ✓ Multithreaded
- ✓ Robust
- ✓ Dynamic
- ✓ Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works:

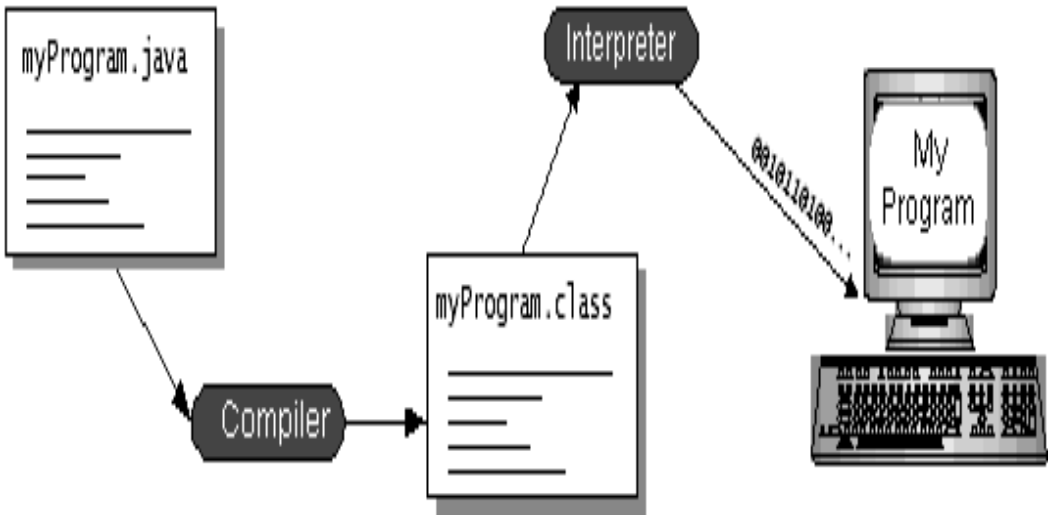

Figure 4.6.1 Working of Java compiler

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program 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.

That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac; illustrated below:

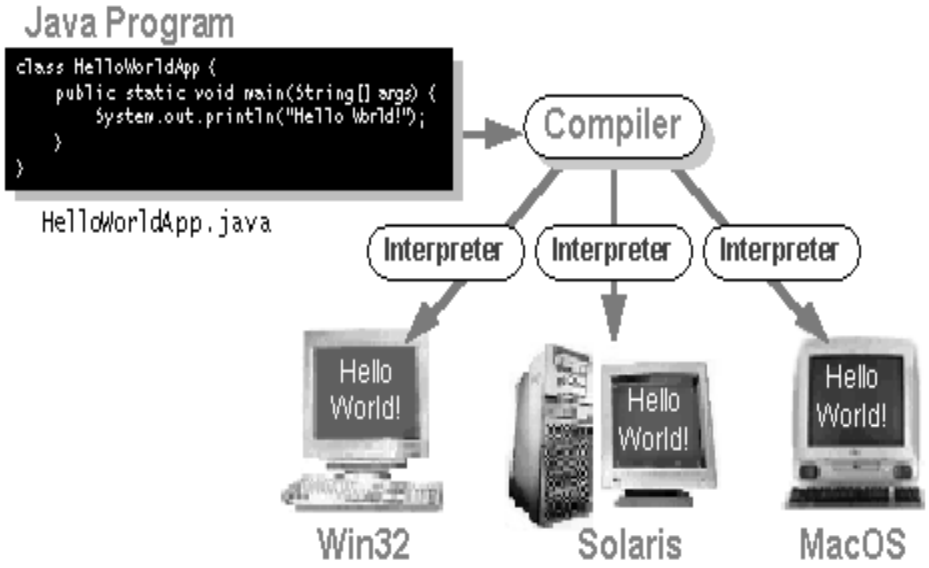

Fig. 4.6.2 JVM on different platforms

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. 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.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

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 of related classes and interfaces; these libraries are known as *packages*.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

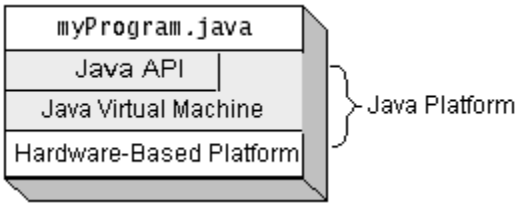

Figure 4.6.3 How Java API and JVM insulate program from hardware

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser. However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side.

Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server. How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform 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 applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) 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 and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans™, can plug into existing component architectures.
- **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.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK :

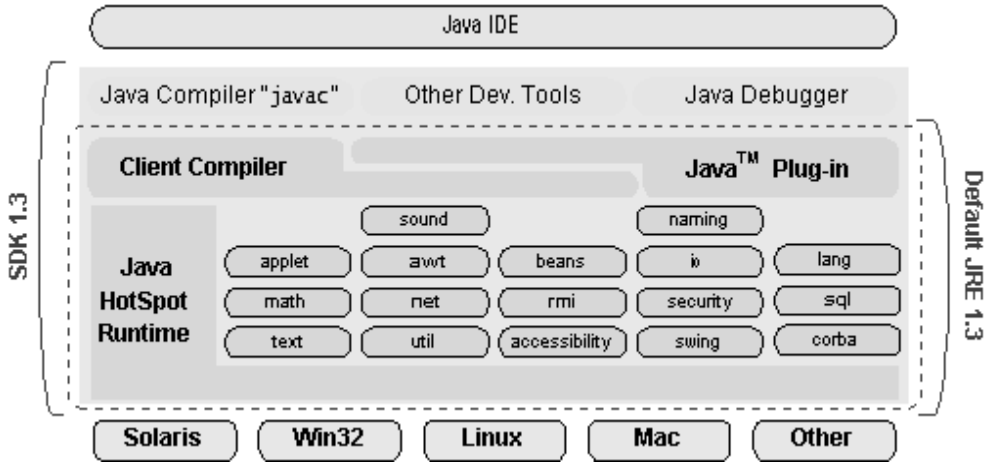

Figure 4.6.4 Java 2 SDK

4.6.2 ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the

choice of the database system almost irrelevant from a coding perspective, which is as it should be.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. The availability of good ODBC drivers has improved a great deal recently. The criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

4.6.3 JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access

mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will give information about JDBC to know what it is about and how to use it effectively.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. *SQL Level API* - The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.
2. *SQL Conformance* - SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the

connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. *JDBC Must Be Implemental on Top of Common Database Interfaces* - The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
4. *Provide a Java interface That is Consistent with the Rest of the Java System* - Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.
5. *Keep It Simple* - This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.
6. *Use Strong, Static Typing Wherever Possible* - Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.
7. *Keep The Common Cases Simple* - Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible. Finally we decided to proceed to the implementation using JavaNetworking. And for dynamically updating the cache table we go for MSAccess database.Java has two things: a programming language and a platform.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

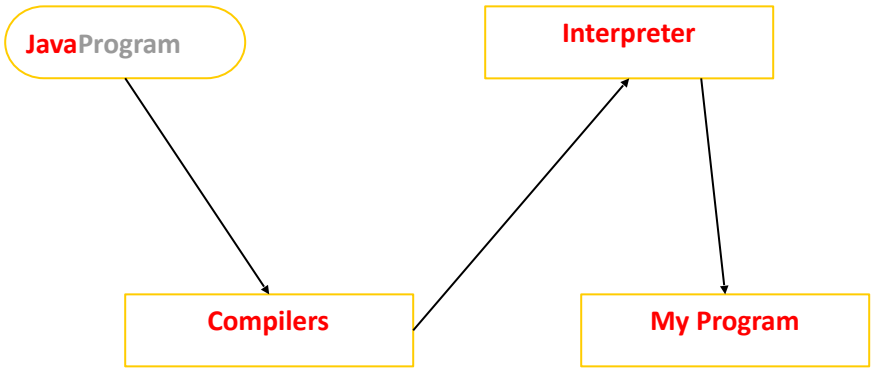

Figure. 4.6.5 Compilation and Interpretation in Java

You can think of Java byte codes 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.

4.6.4 Networking

TCP/IP stack

The TCP/IP is a connection-oriented protocol; and is shorter than the OSI one. The TCP/ IP protocol is illustrated below:

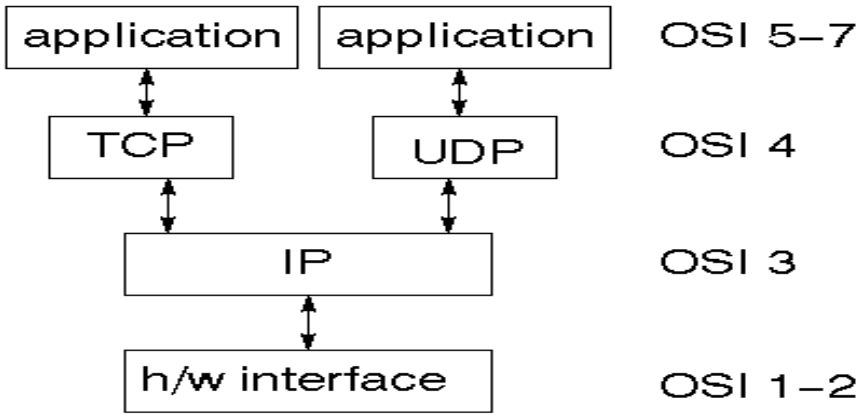

Figure. 4.6.6 TCP/IP Stack Protocol

IP Datagram

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet Addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network addresses

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

Subnet addresses

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.Host addresses - 8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines.

Total addresses

The 32 bit address is usually written as 4 integers separated by dots, as shown below:

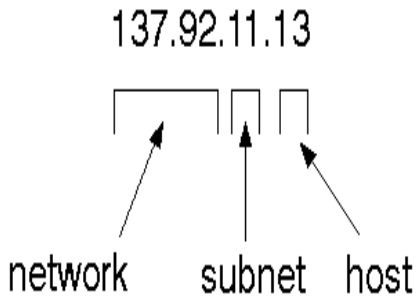

Figure. 4.6.7 32-bit Total address

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

Chapter 5

RESULTS

5.1 Summary

The project results have been as expected. We set out to establish a system that can detect intrusions and block individual intrusions without hampering the working of the rest of the system. The system snapshots follow a user and an administrator as they work on the system.

Initially, the user signs up for services that he requires through an offline channel. Each service is subjected to a certain payment. The cost of using all services is therefore higher than using the cost of a single service. The user uses a password to login and a key to access the services. A double encryption mechanism is therefore used for better security. After login, a user tries to access the services that he has legitimately signed up for, using the valid secret key. If there is an illegitimate access the user's fraud is detected in a fraud database.

The administrator uses his administrator account to log in and monitors the fraud database to detect users who are committing regular frauds on the system. These users are warned. On further misbehavior, the administrator shuts down the user from harming the system by shutting down his computer remotely, to warn him of fraud detection. We also propose future enhancements to enable the administrator to revoke the user's membership if necessary.

5.1.2Screenshots

The following snapshots illustrate the expected output of our project. Snapshots for user and admin login, user and admin interface, etc have been provided. The snapshots show the sequence of activities that the user goes through to finish his task. It also shows the server's role in handling the user's different requests. How the server handles fraud accesses and manages the fraud list to prevent attacks on the system is also illustrated in few of the later snapshots. The snapshots are thus a visual introduction to the system and explain how the system works. The snapshots are presented here in a sequential manner.

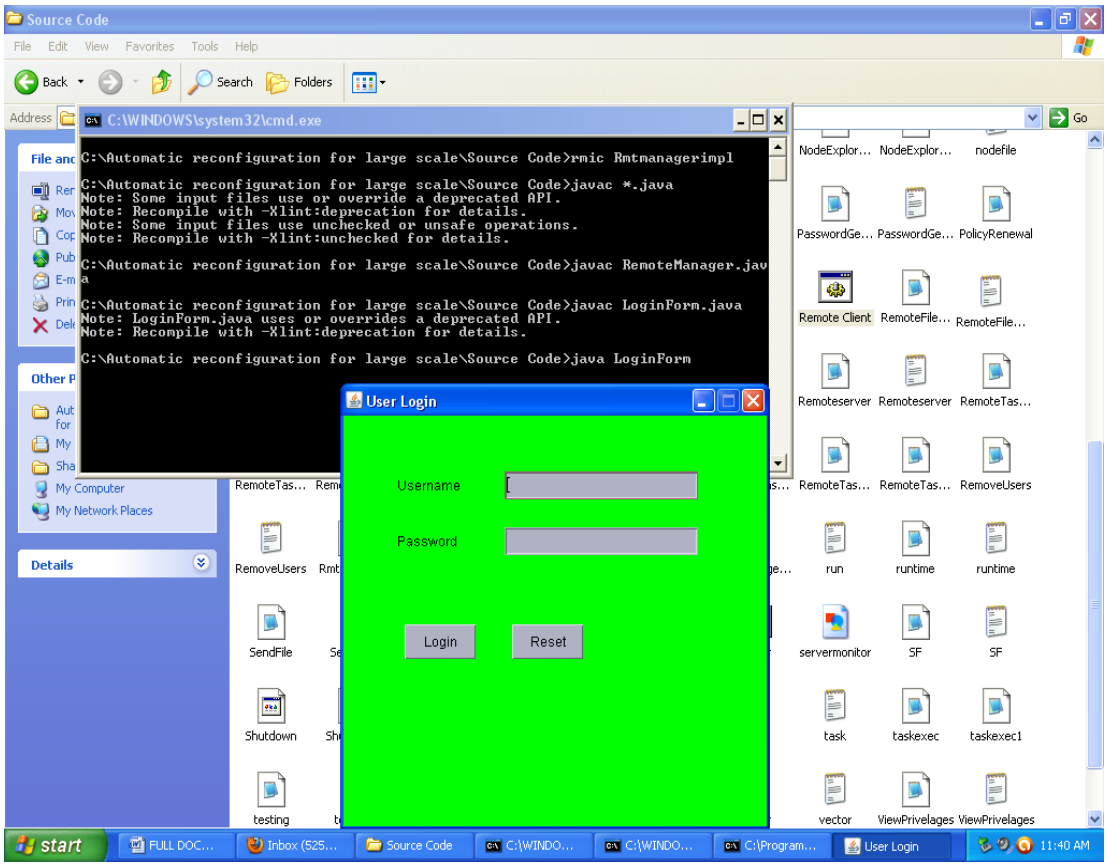

Figure5.1 User Login

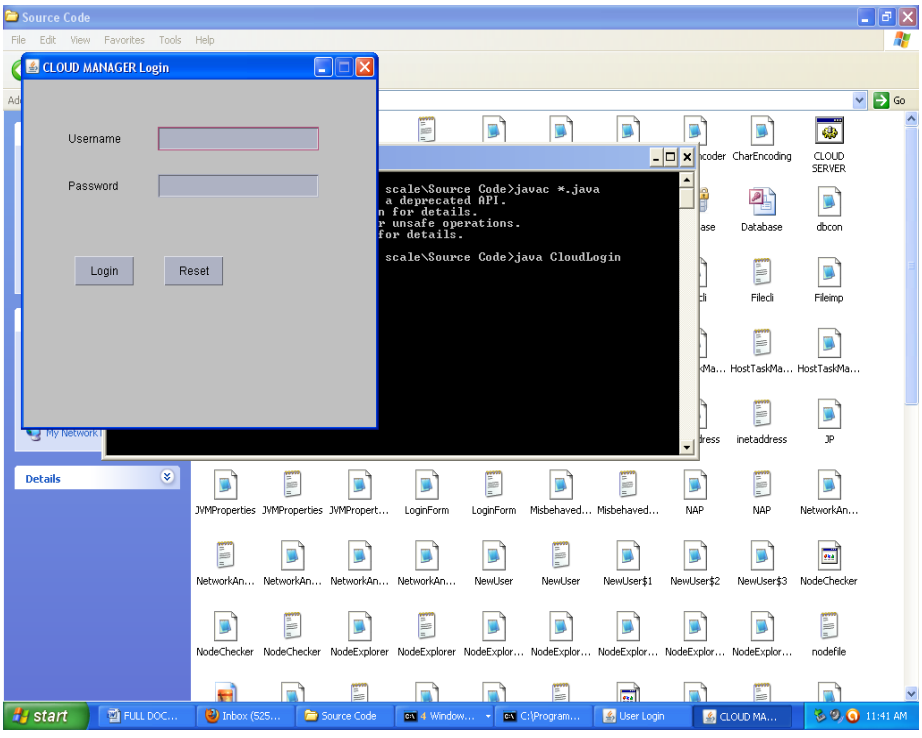

Figure 5.2 Cloud Manager Login interface

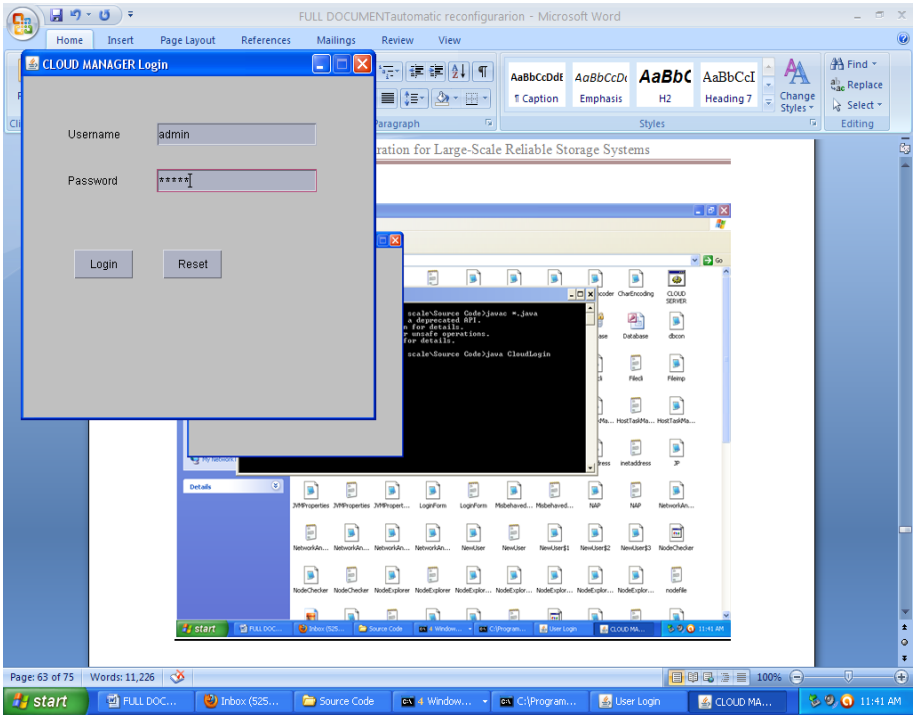

Figure 5.3 Cloud Manager Login

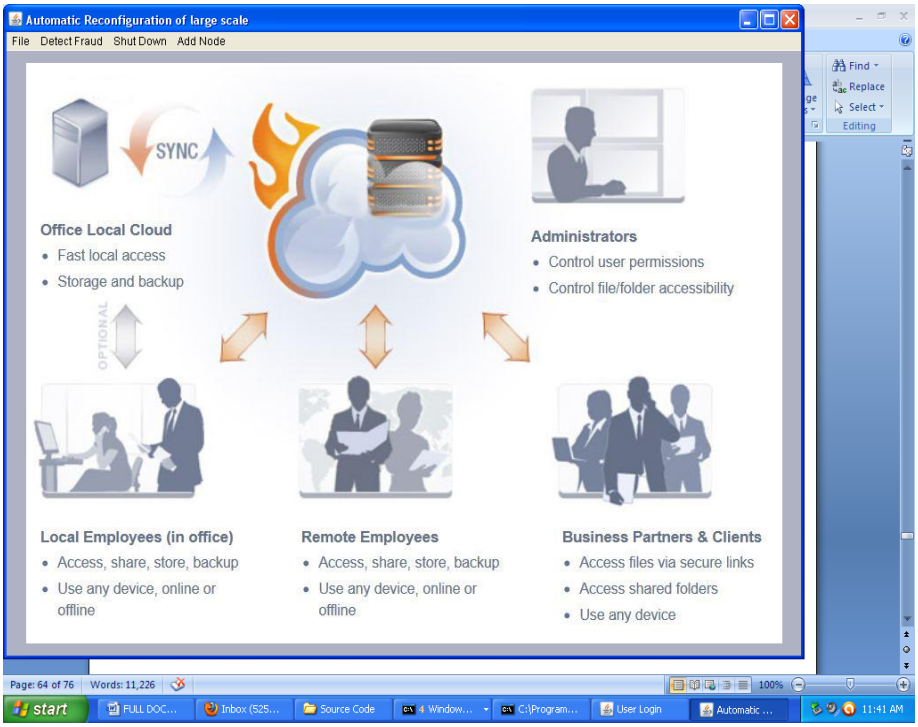

Figure 5.4 Cloud Manager Interface

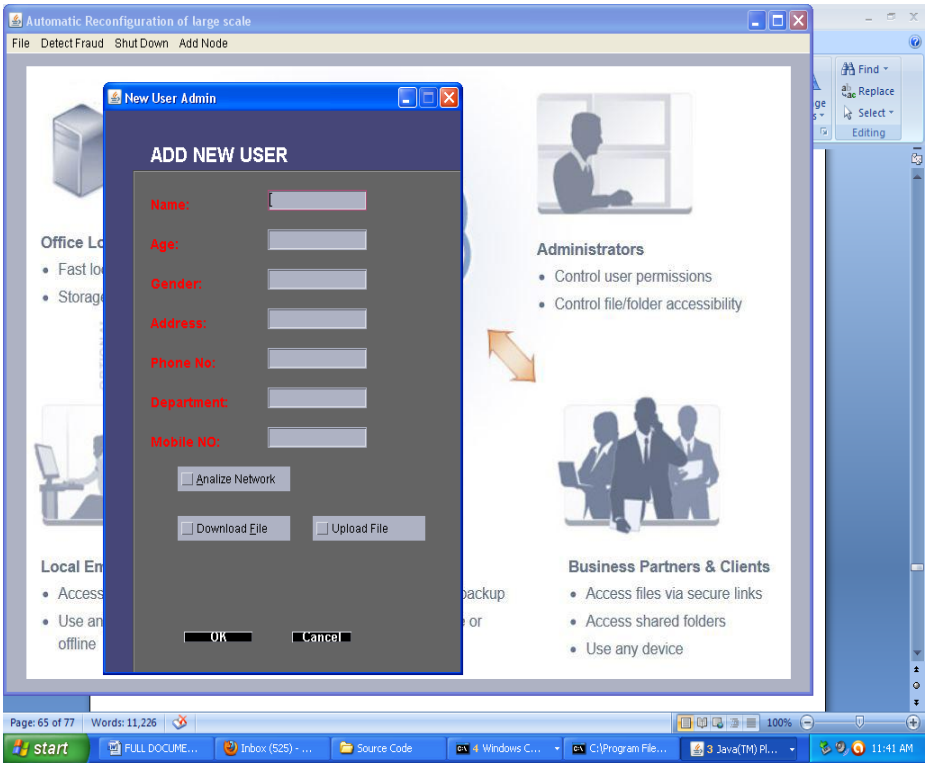

Figure 5.5 Form to add new user

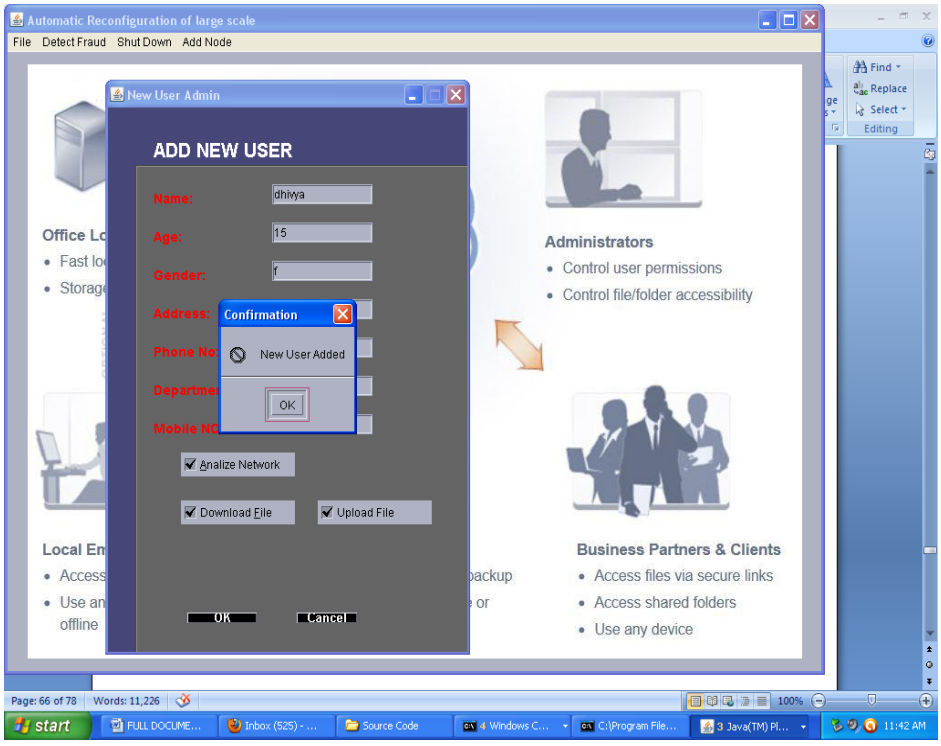

Figure 5.6 Confirmation on adding new user

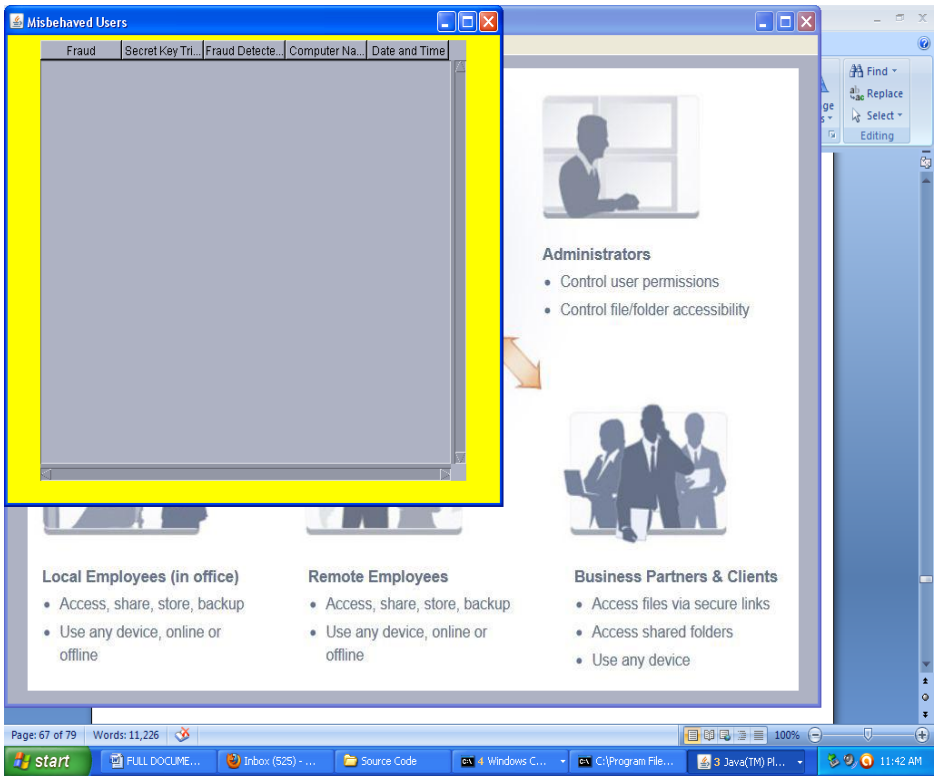

Figure 5.7 Fraud list

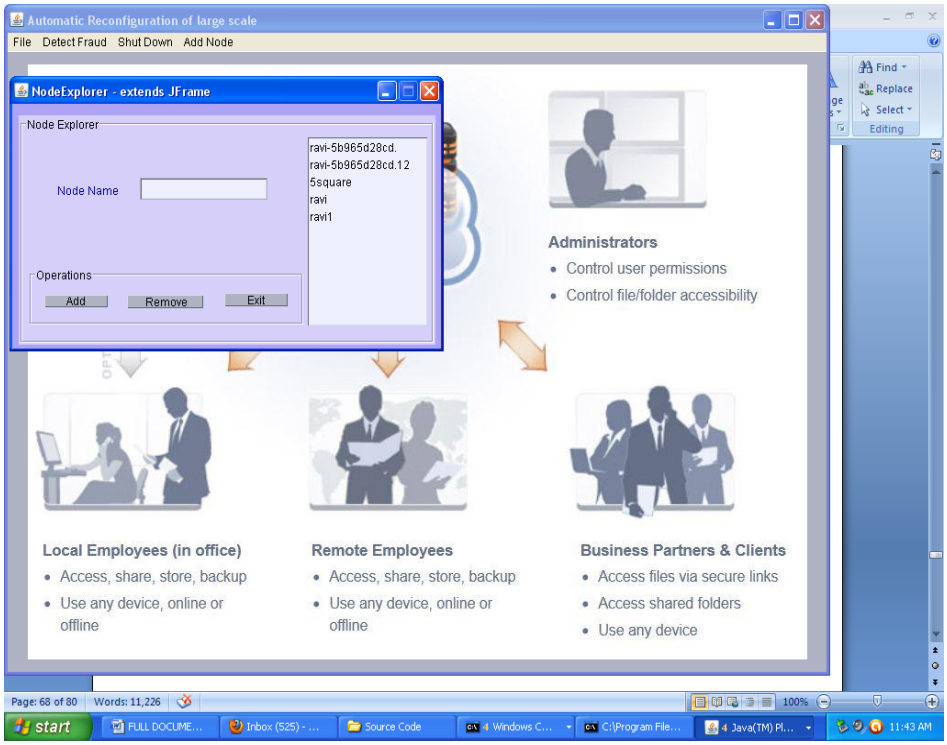

Figure 5.8 Node explorer

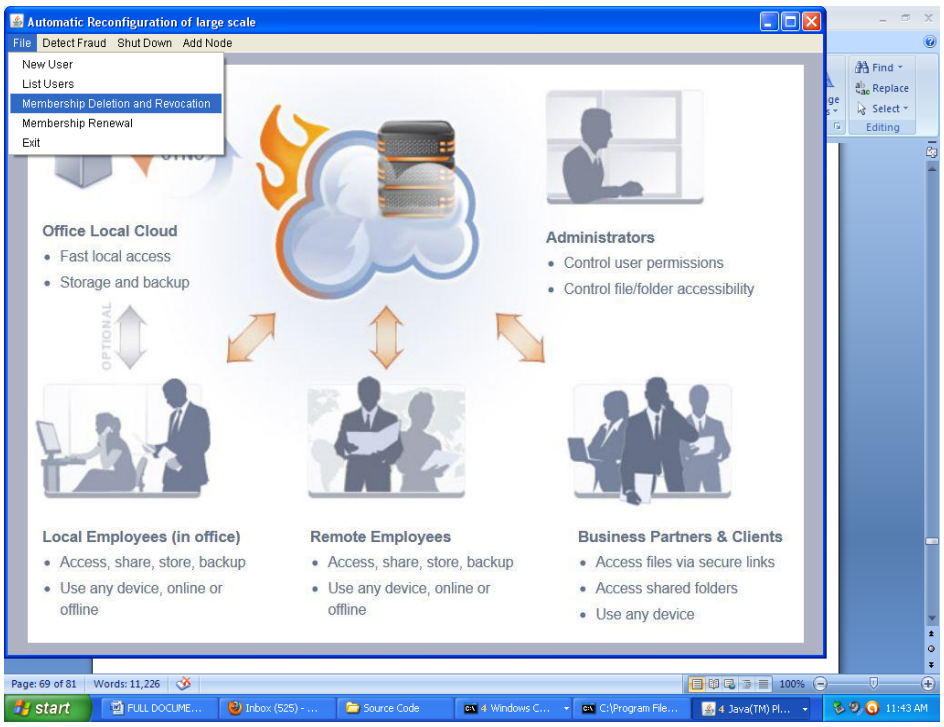

Figure 5.9 Other menu items in admin interface

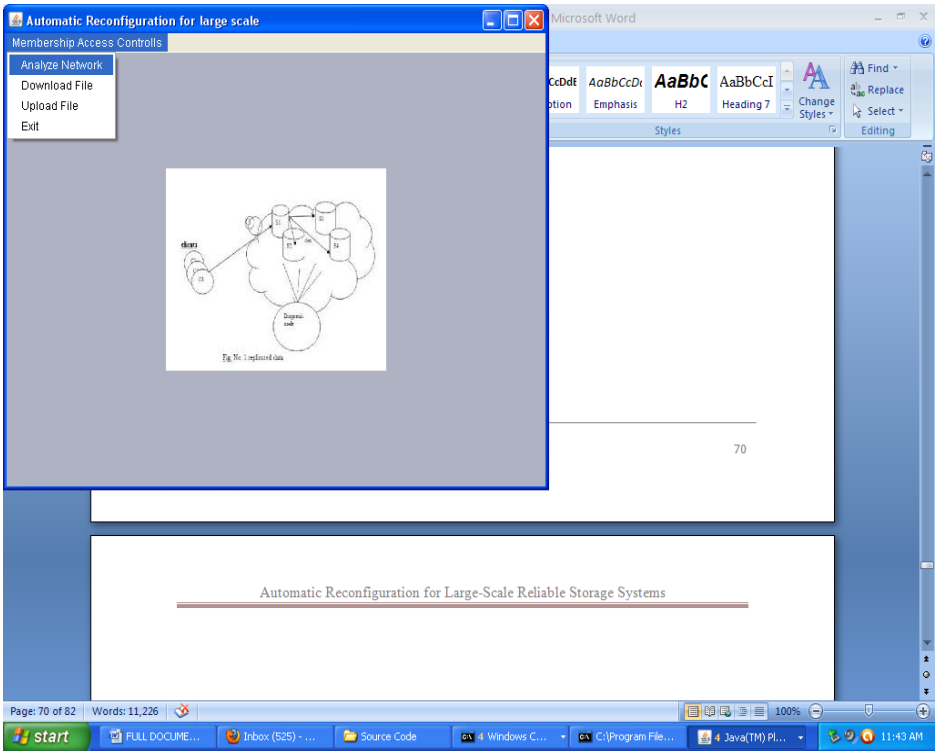

Figure 5.10 Menu items in user interface

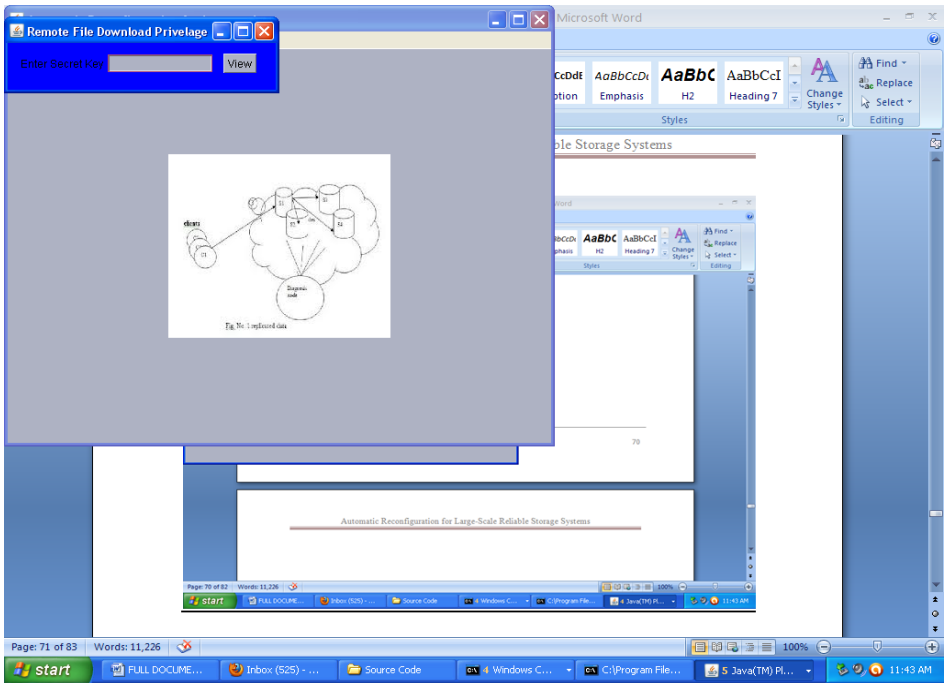

Figure 5.11 Access Privileges interface

5.2 Applications

- Dynamo uses tens of thousands of servers located in many data centers around the world to build a storage back-end for Amazon's S3 storage service and its e-commerce platform.
- In Google's cluster environment each cluster includes an installation of the GFS file system spanning thousands of machines to provide a storage substrate
- The "Practical Byzantine Fault Tolerance" (PBFT) algorithm introduced by Miguel Castro and Barbara Liskov, provides high-performance Byzantine state machine replication, processing thousands of requests per second with sub-millisecond increases in latency. PBFT triggered a renaissance in BFT replication research, with protocols like Q/U, HQ, Zyzzyva, and ABsTRACTs working to lower costs and improve performance and protocols like Aardvark and RBFT working to improve robustness.
- UpRight is an open source library for constructing services that tolerate both crashes ("up") and Byzantine behaviors ("right") that incorporates many of these protocols' innovations.
- One example of BFT in use is Bitcoin, a peer-to-peer digital currency system. The Bitcoin

network works in parallel to generate a chain of Hashcash style proof-of-work. The proof-of-work chain is the key to solving the Byzantine Generals' Problem of synchronising the global view and generating computational proof of the majority consensus

- There is also the BFT-SMaRt library, a high-performance Byzantine fault-tolerant state machine replication library developed in Java. This library implements a protocol very similar to PBFT's, plus complementary protocols which offer state transfer and on-the-fly reconfiguration of hosts. BFT-SMaRt is the most recent effort to implement state machine replication, still being actively maintained.

5.3 Advantages and Disadvantages

5.3.1 Advantages

- It provides the abstraction of a globally consistent view of the system membership. This abstractionsimplifies the design of applications that use it, since it allows different nodes to agree on which servers are responsible for which subset of the service.
- It is designed to work at large scale, e.g., tens or hundreds of thousands of servers. Support for large scale is essential since systems today are already large and we can expect them to scale further.
- It is secure against Byzantine (arbitrary) faults. Handling Byzantine faults is important because it captures the kinds of complex failure modes that have been reported for our target deployments. For instance, a recent report about Amazon’s S3 showed that a bit flip in a server’s internal state caused it to send messages with the wrong content. Additionally, the Byzantine fault model makes it possible to tolerate malicious intrusions where an attacker gains control over a number of servers
- Faulty servers cannot cause our system to behave incorrectly.
- Small attacks cannot overpower critical system functions.

5.3.2 Limitations

- Faulty servers cannot cause the system to behave incorrectly but can degrade the performance of protocols.
- Though the reconfiguration is fast, there is a large variation in the time taken to apply

reconfiguration

- More than one main server and system administrator may be needed if the size of the network is very large. In other words, there is a limit on the scalability of the system.
- These systems are long lived and need to continue to function even though the machines they run on break or are decommissioned. Thus, there is a need to replace failed nodes with new machines; also it is necessary to add machines to the system for increased storage or throughput. Thus, the systems need to be reconfigured regularly so that the network can continue to function.
- Byzantine fault tolerance allows the network to sustain itself even when attacks are carried out by individual systems. Sometimes this may be counterproductive.

Chapter 6

CONCLUSION

6.1 Task

Today, we are more and more dependent on Internet services, which provide important functionality and store critical state. These services are often implemented on collections of machines residing at multiple geographic locations such as a set of corporate data centres. For example, Dynamo uses tens of thousands of servers located in many data centres around the world to build a storage back-end for Amazon's S3 storage service and its e-commerce platform. As another example, in Google's cluster environment each cluster includes an installation of the GFS file system spanning thousands of machines to provide a storage substrate. Additionally, these systems are long lived and need to continue to function even though the machines they run on break or are decommissioned. Thus, there is a need to replace failed nodes with new machines; also it is necessary to add machines to the system for increased storage or throughput. Thus, the systems need to be reconfigured regularly so that they can continue to function.

Our project implements a paper which provides a complete solution for reliable, automatic reconfiguration in distributed systems.

6.2 Achievement

A complete solution for reliable, automatic reconfiguration in distributed systems should achieve following properties:

- **Scalable**

  Support for large scale is essential since systems today are already large and we can expect them to scale further.

- **Secure against Byzantine faults**

  Handling Byzantine faults is important because it captures the kinds of complex failure modes that have been reported for our target deployments. Additionally, the Byzantine fault model makes it possible to tolerate malicious intrusions where an attacker gains control over a number of servers.

- **Consistent**

  It provides the abstraction of a globally consistent view of the system membership. This abstraction simplifies the design of applications that use it, since it allows different nodes to agree on which servers are responsible for which subset of the service.

6.3 Final Outcome

This paper presents a complete solution for building large scale, long-lived systems that must preserve critical state in spite of malicious attacks and Byzantine failures. We present a storage service with these characteristics called dBQS, and a membership service that is part of the overall system design, but can be reused by any Byzantine-fault tolerant large-scale system.

The membership service tracks the current system membership in a way that is mostly automatic, to avoid human configuration errors. It is resilient to arbitrary faults of the nodes that implement it, and is reconfigurable, allowing us to change the set of nodes that implement the membership service when old nodes fail, or periodically to avoid a targeted attack. When membership changes happen, the replicated service has work to do: responsibility must shift to the new replica group, and state transfer must take place from old replicas to new ones, yet the system must still provide the same semantics as in a static system.

Chapter 7

SCOPE FOR FUTURE WORK

7.1 Developing a completely remote network

We propose to work on the following future modules in due course of time to improve our model's features and capabilities.

Network Analysis Module

Network analysis module provides the user with an ability to know the active and inactive desktops in the network, and then the user is provided with an option to select the active desktop and execute the task as intended.

Remote File Manager Module

In this module both the users can perform file upload and download task based on their access privileges.

Send File Module

The user can send files to the remote nodes simply by specifying the name of remote node.

Checking Remote Platform Module

A list of active desktops is provided to the user who can then select a particular desktop to display the name of the operating system installed in the remote desktop.

Shutdown Remote Node Module

The user is provided with an option to know the kind of processes that the remote desktop is executing, if the remote user is found malfunctioning then that desktop is turned off by the remote manager by setting the time in minutes.

Adding New Nodes Module

The user also has a provision to add new desktops to the network using the name of the remote desktop.

7.2 Tracing Fraud Users in Mobile Devices

A technique called cubix technique is used to cause the notorious problem common in P2P communication systems, also called free-riding, where some peers take advantage of the system by providing little or no service to other peers or by leaving the system immediately after the service needs are satisfied. Peer cooperation is thus the fundamental requirement for P2P systems to operate properly. Since peers are assumed to be selfish, incentive mechanisms are necessary to promote peer cooperation in terms of both cooperativeness and availability. Typical incentive mechanisms for promoting cooperativeness include

- Reputation based approach - In this approach, peers are punished or rewarded based on the observed behavior. However, low availability remains an unobservable behavior in such systems, which hinders the feasibility of the reputation-based mechanism in improving peer availability.
- Payment-based approach - The payment-based approach provides sufficient incentives for enhancing both cooperativeness and availability, and thus, is ideal to be employed in multihop uplink communications among peer clients in our system.

BIBLIOGRAPHY

Websites

http://java.sun.com

http://www.sourceforge.com

http://www.networkcomputing.com/

http://www.roseindia.com/

http://www.java2s.com/

Papers

- "Automatic Reconfiguration for Large-Scale Reliable Storage Systems", Rodrigo Rodrigues, Barbara Liskov, Kathryn Chen, Moses Liskov, and David Schultz, IEEE 2012.
- "Antiquity: Exploiting a Secure Log for Wide-Area Distributed Storage", Hakim Weatherspoon, Patrick Eaton, Byung-Gon Chun and John Kubiatowicz, European Conf. Computer Systems (EuroSys '07).
- "Authentication and Integrity in Outsourced Databases", EinarMykletun, Maithili Narasimha and Gene Tsudik, ACM 2005.
- "Secure routing for structured peer-to-peer overlay networks", Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach, Proc. of the 5th Usenix Symposium 2002.