A Dissertation

ON

Fault Tolerant Distributed File System

Submitted in partial fulfillment of the award of

MASTER OF TECHNOLOGY

Degree in

SOFTWARE ENGINEERING

By

KESHAV GUPTA

(2k11/SWE/07)

Under the guidance of

MS. DIVYASHIKHA SETHIA

2013
Department of Software Engineering,
Delhi Technological University
Delhi-110042

DECLARATION

I hereby declare that the thesis entitled **Fault Tolerant Distributed File System** which is being submitted to the Delhi Technological University, in partial fulfillment of the requirement for the award of degree of Master of Technology in Software Engineering is an authentic work carried out by me.

Keshav Gupta

2K11/SWE/07

Department of Software Engineering

Delhi Technological University

Delhi – 110042

CERTIFICATE

This is to certify that the thesis named **Fault Tolerant Distributed File System** submitted by Keshav Gupta, roll number 2K11/SWE/07, in partial fulfillment of the requirements for the award of degree of Master of Technology in Software Engineering, with researchers' work duly referenced has been carried out under my supervision.

Ms. Divyashikha Sethia
Assistant Professor
Department of Software Engineering
Delhi Technological University
Bawana Road, Delhi- 110042

ACKNOWLEDGEMENT

I am glad to take this opportunity to thank my project guide Ms. Divyashikha Sethia for giving me the chance to work in this research area along with her relentless support and encouragement. It would not be possible to make this dissertation on run without her consistent guidance and valuable time, data, and reference, to name a few of her contributions.

I am also thankful to Head of Department Dr. Daya Gupta and all the faculty members, my classmates and of course all the researchers' active in this field for their direct or indirect support and motivation during the work.

Keshav Gupta
2K11/SWE/07
M. Tech. (Software Engineering)
Delhi Technological University, Delhi- 11004

ABSTRACT

Despite distributed file systems having already been implemented for individuals and companies, users continue to face problems in using these systems and it is a still a big challenge for the researchers. The functionalities of a well functioned and an ideal distributed file system would be to provide high scalability, provide shared access to the same set of files to all its users, provide high performance, provide more storage space and more user friendliness. Google File System answers to these problems in a very efficient way and been a source of inspiration in development of this project. We are trying to simulate working of GFS for small scale data set.

In this project we have implemented a Distributed File System with multiple levels of fault tolerance. At first level we are trying to filter faulty nodes/clients before they enter into system or we can say register with the server. At second level, to avoid failure we are maintaining chunk replicas so that if a client possessing file chunks gets disconnected, the file can still get downloaded.

For security purpose, DES algorithm has been used for encryption and decryption. So we are successfully distributing and retrieving a file from a DFS in a very secure and fault tolerant manner. Thus, helping in load balancing by distributing the file over the network and increasing the overall system performance. Many different concepts such as transparency, fault tolerance, scalability, reliability have been defined in this dissertation and solved them in FTDFS approach. Various implementations and design strategies have been demonstrated by a survey along with different distributed systems for file sharing. Testing and performance evaluation of the system have also been carried out in this dissertation and also possible future work or the improvements have also been discussed.

LIST OF FIGURE(S)

LIST OF TABLE(S)

ACRONYMS

DES	Data Encryption Standard
DFD	Data Flow Diagram
DFS	Distributed File System
FTDFS	Fault Tolerant Distributed File System
GFS	Google File System
GUI	Graphical User Interface
I/O	Input/output
IP	Internet Protocol
LAN	Local Area Network
NFS	Network File System
TCP	Transport Control Protocol

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

These days, more resources are required by new applications than what are actually available on an inexpensive machine and problems are faced with the business processes by the organizations that are no longer fit on single cost effective machine. Either at home or business computers, there is certain amount of data stored which is needed to be kept private as the data could be sensitive. The organizations would not be able to run properly or would face a high loss in the terms of cost and effort. A small example could be a student's laptop containing student's school assignments or lecture notes. Loss of such data would result him in redoing all the assignments and making the lecture notes from the start. Here comes the use of the Distributed File System (DFS), which helps businesses and individuals in keeping the data or the information important to them in a safe and secured way.

Today, everyone know about one of the most popular and most used websites such as Google or YouTube, also the social networking phenomenon called Facebook. There are many significant uses of these websites. Google nowadays isn't only a search engine but it supports other functionalities such as uploading pictures and videos. One can create an email account provided with a few gigabytes of space. Other fascinating and attractive features introduced by Google are Google Drive, Google Maps and also Google Earth applications. All these services are pretty efficiently provided by Google and are heavily data intensive. Here comes a big question to be asked that how big companies like Google, YouTube or Facebook do it. The answer to the question above is that these big companies use Distributed File System in order to provide their services on a high level.

In the creation of the Fault Tolerant Distributed File System (FTDFS) to provide an environment where users can distribute the files among each other and the files are stored in an safe and secured way, many technical challenges were faced which are discussed in this dissertation. In order to create a great design of a FTDFS, it is important to be familiar with the present Distributed File Systems and also should have a good understanding of server/ client based architecture. Since Java is the programming language used to build this DFS, it is important to know the characteristic of java network and object-oriented characteristic.

1.2 Motivation

A file system is the method of permanent storage of files and organizing those files so that it is easy to find those files and access them. A distributed file system is conversely a network of file systems, which provides the storage of the files among the commodity of connected computers. A typical configuration of a DFS is a collection of workstations and mainframes connected by a local area network (LAN). A DFS is implemented as part of the operating system of each of the connected computers. One of the main drawbacks found in the prevailing file systems is the lack of the scalability. Security and reliability is quickly becoming a mandatory feature of data storage systems. Some of the key features of this project are Manipulation of files using Framework, Improved Scalability, and Improved reliability (Using replication), Security (Using cryptography), Authentication and High Performance.

1.3 Aims and Objectives

The main aim of the project is to provide a Fault Tolerant Distributed File System (FTDFS). The DFS provides storage for secure files, high scalability, local independence and transparency that is the migration capabilities for data.

The users are able to upload files. The uploaded file is to be first encrypted to increase security.

After encryption file will be divided into various chunks depending on size of file based on given chunk size. The chunks are also replicated to increase reliability of the system. The chunks created and replicas will be distributed among all the clients registered provided original chunk and its replica will not be stored on same client. After a successful upload a key will be generated and will be sent to the client which will be named including client ID and name of the file uploaded. While downloading the file, client will send this key to the server. Server will get client ID from the key, directly go to the path specified by the clients where chunks are stored and fetch the chunks related to the file. If a particular chunk is missing or client is unavailable, replicated chunks are retrieved. After that all the chunks are merged together with the help of a Batch file which was created during the creation of chunks. The retrieved file will be decrypted and will be sent back to the client. Avoid problems such as security problems, bad file replication, poor scalability and slow performance. Evaluate the project on the basis of scalability by connecting a very high number of nodes. Also checking its reliability in the situations when 1 or more client exits.

More future enhancements could be done such as multiple servers and use of more powerful and advanced algorithms and techniques for effective file replications and retrieval.

1.4 Thesis Outline

Chapter 1 introduces Distributed File System along with motivation, Aims and objectives of this project. **Chapter 2** presents Literature review related to distributed file systems. **Chapter 3** describes the Requirement analysis and Feasibility Study in Detail. **Chapter 4** presents the system architecture of current systems and FTDFS in detail. **Chapter 5** presents the Implementation part of the project along with system testing and performance study. At last in **Chapter 6** we are summarizing thesis with Conclusion and Future Work.

CHAPTER 2

LITERATURE REVIEW

An extensive review of research was carried out in order to gain better understanding of the Distributed File System and the tools and technical skills required to implement the project requirement.

2.1 Concept

DFS is a system that supports the sharing of information through a network in the form of files. A file service which is very well designed provides access to the stored files at a server with the performance and reliability similar to that of the disk. In other words it can be said that a DFS allows the users to store and access remote files like a local computer would, but from any computer in the network. It has a capability of distributing the files from one computer to other computers. If it is to be said in scientific terms, a Distributed File System is a distributed implementation of the classical time sharing model which is constructed on a file system where it allows one or many users to share the files and storage resources [2]

DFS supports the common operations effectively on the same kind of sharing resources. It supports especially the operations for the files physically dispersed among the different sites of a distributed system. If it is to be seen from the inheritance point of view, a DFS is a system which is inherited from the File System and Distributed System. A DFS provides the common files operations such as read, delete, rename, share and copy. These operations are not only used in the certain local sites of DFS but they are also used in other sites through authorization access. A DFS can be seen as a polymorphic subclass which accommodates its processors or the resources distributed mechanism that is derived from Distributed System [2].

A Distributed Operating System can be seen as a collection of interconnected computers which appears to the users or the clients as a single system. If any system doesn't work, or crashes, it won't affect the working of the other systems interconnected. It manages the data i.e. the files and migrate the data/file from one site to other sites.

2.2 Fault tolerance in DFS

There has been a lot of research work done recently for improving the fault tolerance of Distributed Systems. Alexandru Costan and others provided a Fault Tolerance Approach for Distributed Systems Using Monitoring Based Replication. They proposed an approach relying on replication techniques and based on monitoring information to be applied in distributed systems for fault tolerance. Their approach uses both active and passive strategies to implement an optimistic replication protocol [3]. Marieta Nastase and others proposed a Fault Tolerance scheme using a Front-End Service for Large Scale Distributed Systems which is based on a set of replicated services running in a fault-tolerant container and a proxy service able to mask possible faults, completely transparent to a client [4]. Sunil Chakravarthy, Chittaranjan Hota proposed a Secure Resilient High Performance File System for Distributed Systems that has the potential to cope with increasing number of participants and ensures strong file encryption [5]. Swastisudha Punyatoya gave Generic Algorithm-Based Fault Diagnosis for Distributed Systems which results in decrease in time and no. of messages passed in order to diagnose the fault [6]. Ioan Petri, proposed a Quorums Systems as a Method to Enhance Collaboration for Achieving Fault Tolerance in Distributed Systems, 2009[7]. Bharath Balasubramanian gave a Coding-Theoretic Approach for fault tolerance in Distributed systems which uses a fusion based fault tolerance scheme [8].

2.3Replication in DFS

Bin Cai, Changsheng Xie, Guangxi Zhu proposed an Effective Distributed Replication File System for Small-File and Data-Intensive Application [9]. It works with a single metadata server and multiple storage nodes, deploys whole-file replication scheme at the file level, and tracks what storage node a file is replicated on. Anna Hat, Xiaowei Jin, Jo-Han So0 provided a survey about Algorithms for File Replication in a Distributed System which included File Replication by Using a Single-Host Sequential Update Algorithm, File Replication by Using a Single-Host Concurrent Update Algorithm, File Replication by Using a Multiple-Host Update Algorithm and Algorithm in the System without File Replication[10]. Yan Chen, Randy H. Katz and John D. Kubiatowicz gave a Dynamic Replica Placement policy for Scalable Content Delivery [11]. Gyuwon Song, Suhyun Kimz and Daeil Seo also proposed a Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems[12] in which they develop a replica placement algorithm which Exploits the availability pattern of each individual peer.

2.4 Current System Analysis

Today Sun's NFS, Google File System and Hadoop DFS are the most commonly and widely used systems.

2.4.1 Sun's NFS

2.4.1.1 Overview

Sun Microsystems came with an idea of the Distributed File System in developing their NFS that stands for Network File System for Unix-like Operating System. This DFS is mainly for the local area networks i.e. LAN. Under UNIX, in this distributed system, it provides a transparent access to the remote files to the clients. But today, implementation of the network file system can be applied to almost all the operating systems even including the non-Unix systems. Some examples of such systems are MS-DOS, Microsoft or Macintosh. Sun developed an open protocol that specified the exact message formats that server and clients would use to communicate. Using this approach different groups develop their own NFS servers. Companies such as IBM or NetApp are selling such NFS servers.

2.4.1.2 Sun's NFS Architecture

Talking about the protocol, the Network File System provides a set of Remote Procedure Call that is RPC for remote file operations. It looks up for a file in the directory viewing all the files as a root providing the transparent access to all the directories and files, manipulates the directories and links, clients have an access to create and remove files, getting and setting file attributes like rename, delete, write. The clients are given the access to read and write files.

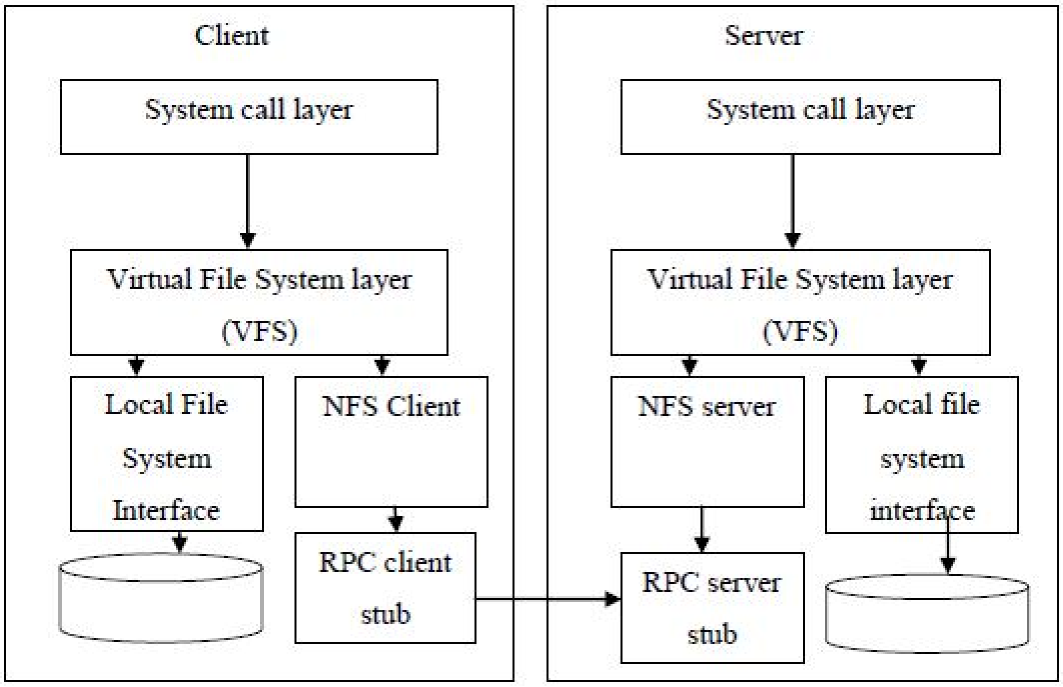

Client
System call layer
Virtual File System layer (VFS)
Local File System Interface
NFS Client
RPC client stub
Server
System call layer
Virtual File System layer (VFS)
NFS server
Local file system interface
RPC server stub

Figure 2.1: The Sun NFS Architecture [13]

2.4.1.3 Usability

There are some advantages of Sun's NFS that come along with some disadvantages as well. Some of the advantages of Sun's NFS are Scalability, Access transparency, Hardware and OS heterogeneity and File Replication. These can also be thought to be the main goals of the Sun's NFS. Despite having a lot of advantages, it has also has some disadvantages. Mobile devices are not fully supported by NFS since migration transparency is not fully achieved. It doesn't support the file replication with updates. It does not enforce a single network; therefore each client sees the file system their way. Hence, location transparency is affected. NFS uses the underlying UNIX file protection mechanism on servers for access checks. Each RPC request from a client conveys the identity of the user on whose behalf the request being made. The server temporarily assumes this identity, and the file accesses that occur while servicing the request are exactly checked as if the user had logged in directly to the server.

2.4.2 Google File System

2.4.2.1 Overview

This file system was developed by Google for its data storage and other needs. Here nodes are divided into chunk servers and 1 master node. Every file is divided into fixed size chunks which are stored at chunk servers in replicated manner where each chunk is identified by a chunk handler. The master node maintains all the metadata and manages all other nodes.

2.4.2.2 GFS Architecture

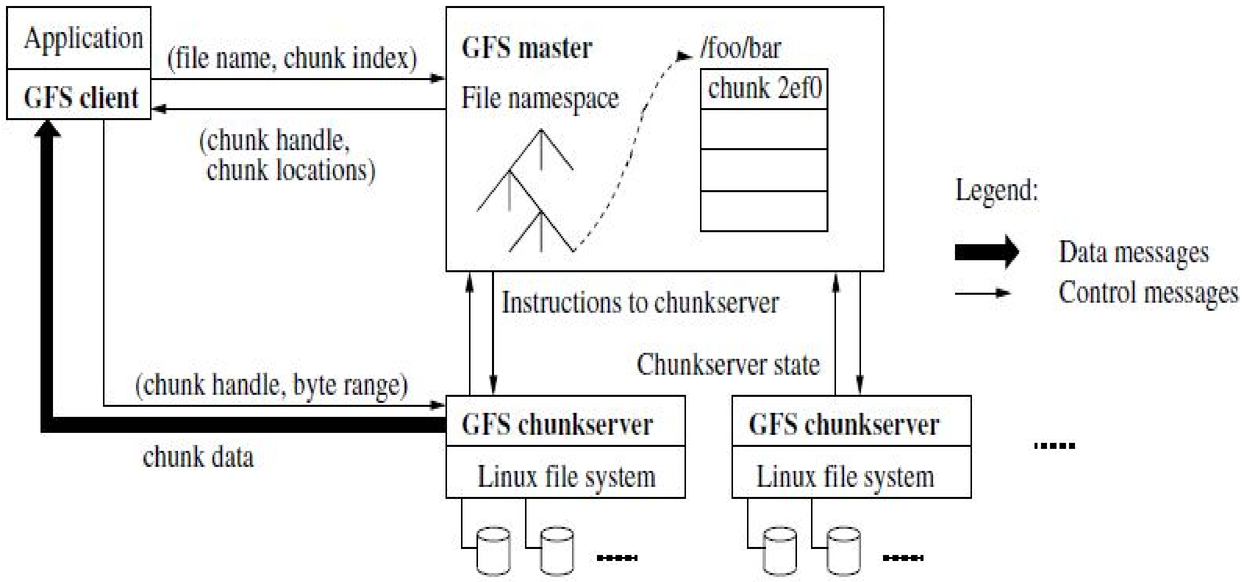


Figure 2.2: Google File System [1]

2.4.2.3 Usability

There are both advantages and disadvantages of GFS. Its advantages include high availability with triple replication and intelligent replica placement. It delivers a very high performance while maintaining very low cost. The management is also autonomous in nature. Talking about the disadvantages it is suitable for data centers for searching workloads but not for small read/write operations. Also, it is not open source software yet.

2.4.3 Hadoop

2.4.3.1 Overview

HDFS stands for Hadoop Distributed File System, provides framework from the analysis and transformation of very large data sets. Executing application computations in parallel close to their data, computation across many hosts and portioning of data are some of the important characteristics of Hadoop. Regulation access to files by clients and management of the file system namespace is managed by the Hadoop Architecture components which are NameNodes and a master server. The storage attached to the nodes is managed by DataNodes. The file which is allowed for the storage by the file system namespace in internally split into one or more blocks which are stored in a set of DataNodes. Operations such as renaming of files and directories, opening and closing of files are executed by the NameNode. The read and write request from the client is served by the DataNodes. DataNodes are also responsible for deletion, creation and replication upon instruction from the NameNode[14].

2.4.3.2 Hadoop Architecture

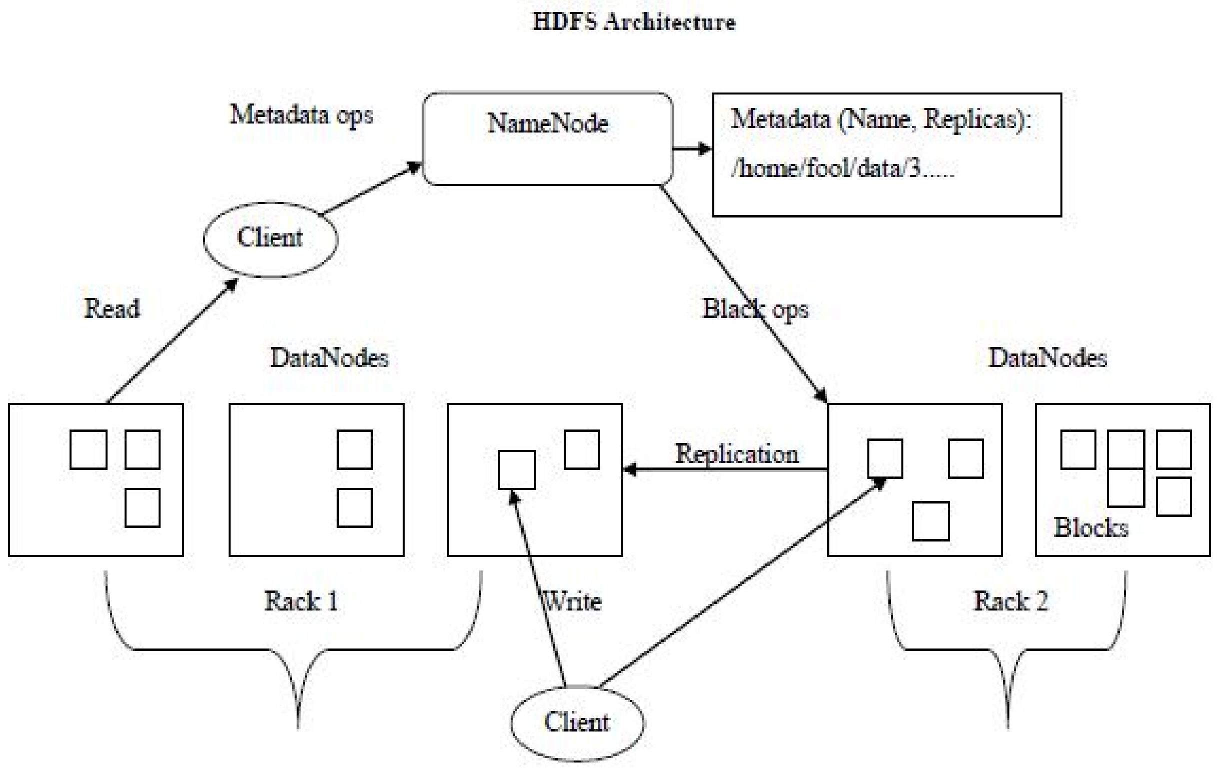


Figure 2.3: The HDFS Architecture [14]

2.4.3.3 Usability

Similarly, there are some advantages and disadvantages of Hadoop DFS. Starting with the advantages first, HDFS is designed for storing large files with streaming data success patterns, running on clusters of commodity hardware. The files with sizes in megabytes, gigabytes or terabytes can be operated. The pattern on which the HDFS is build is the data processing pattern which is writing-once, read-many time's pattern. Expensive hardware is not required to run it. It also has some disadvantages. It does not cover all the requirements although it implements much of the functionality that our platform requires. It was originally designed for the I/O bounds in which reading, writing and transferring data are the most time consuming operations; hence it is not conservative in CPU utilization and memory usage. The technologies of Hadoop are not very well suited for mobile devices.

2.4.4 Conclusions

Sun's NFS is built around the idea of simple and fast recovery in the face of server failure. It achieves this end through careful protocol design. NFS does so in a slightly ad hoc fashion which can result in observably weird behavior. Secondly, also studying in depth about the HDFS (Hadoop Distributed File System), I concluded that it is highly reliable and provides high scalability. It can also run more than once processes at that same time and can store massive amounts of data with a very high fault tolerance. GFS is designed according to the requirements of Google and is suitable for large data sets. Also it is not open source yet.

CHAPTER 3

REQUIREMENT ANALYSIS AND FEASIBILITY STUDY

3.1 Requirement Analysis

A simple statement of requirement can be said that what a system must be able to do and what characteristics it must possess. It can be divided into two types of requirements which are user requirements which can sometimes also be called as business requirements as it mainly focuses on user needs and system requirement. User requirements are more technical as they describe the implementation of the system.

3.1.1 FTDFS User's requirements

Accuracy: The stored directories or files are updated in real time in FTDFS. It ensures that communication users can access to the current shared data while communicating.

Security: Each user has to register in order to become a client and proceed with the data sharing. So to login, they need to provide their correct username and password. What makes the security even higher is that the file with user's login details never leaves the local machine.

Reliability: Even if a client with file data disconnects, the client should be able to download file. Also, only clients with high performance measures should be able to register with the server.

Communication: All the users who are familiar with different operating environments such as Windows, UNIX. It can be set up in most of operating systems.

Error Indication: If a user makes any mistake on file operation in FTDFS, there will be a dialog box with system error message will appear so that the user can correct that operation.

Friendly GUI: Readability and Friendliness are the two main issues on which the Graphical User Interface of FTDFS's design concentrates. All the file operations are easy to deal with and operate through the GUI components. User also gets to enjoy the local shared data and remote communication by friendly platform with other computers in FTDFS.

Flexible Access: Accessing the remote shared files and directories is as easy and flexible as on a local machine. So users are hardly aware that the file or directory that they access is on a remote machine in FTDFS.

3.1.2 FTDFS System requirements

The requirements are usually called System requirement from the developer's perspective. There is no obvious distinction dividing a user requirement and a system requirement. There are two types of system requirement viz. Functional requirement and Non Functional requirement.

3.1.2.1 Functional Requirements

Share: On a local machine in FTDFS, a file or a directory operation can be executed. The use is for selecting directory to be shared folder of the shared resource on local machine.

Upload: A file can be uploaded by the distributed client on the server in the LAN network.

Encrypt: The file once uploaded and sent to the server is encrypted by the server using DES Algorithm.

Split: The uploaded file after getting encrypted is spitted into chunks before being sent to the other clients connected to the server.

Distribute: The server distributes those chunks of encrypted data among the clients.

Download: Once the request to download a file is sent by the client who uploaded the file on to the server, the server sends a random key to that client. With the help of that key, the original file can be downloaded.

Merge: Once the key is selected by the user, the key merges those chunk sizes of data which were distributed among several clients back together and forms the original copy of that file.

Decrypt: Once the data is merged and the file has been re-created, but in encrypted format, the key decrypts the file before sending it to the client who requested for that file download.

Path of resource: On local machine, system feature is located which holds the responsibility for displaying the logical path of remote shared and local shared resources.

File Permission: A system feature is addressed in local machine which is capable of showing the permission of the remote shared file or directory at present.

3.1.2.1 Non-Functional Requirement

Operational requirements, performance requirements and security requirements are the three types of Non-Functional requirement since they reflect the behavioral characteristic that a system must have.

Operational Requirements

FTDFS will operate in Window, UNIX etc.
FTDFS is capable of sharing the files of different formats and large sizes such as Word documents, HTML, Excel tables, PDF files etc.

Performance Requirements

In FTDFS, the response will be sent back immediately when local machine/server machine receives a request from the remote machines/clients.

Security Requirement

Every registered client has their own unique username and password through which they can be connected to the server.
The uploaded files on the server are protected as they get encrypted and split among the clients and kept safely and securely.

3.2 Feasibility Study

A feasibility study includes a detailed assessment of the need, value and practically of a proposed system development. In Software Engineering, feasibility analysis is used to define the system and its business requirements. It also determines the vital risks connected with the project that must be referred. Distinct project request decides distinct process and format for the feasibility analysis, in our system, it comprises of two techniques which are Technical feasibility and Operational feasibility.

3.2.1 Technical Feasibility

Technical feasibility is the first technique in the feasibility analysis of FTDFS. It basically focuses on the extent, which FTDFS can be successfully designed, developed and implemented. For this purpose, it is mandatory to examine the technical characteristic of FTDFS.

- Following the trace of conventional client server DFS, FTDFS realized the communication along clients with server in network.

- FTDFS will be advanced on Java platform with Java's object-orientated and network-orientated application techniques.
- On the basis of conventional DFS, FTDFS will be applied on distinct operating environment, such as Windows, UNIX. Etc.
- FTDFS also realizes portability which is one of the most important characteristic of software engineering design. From installation to implementation, it ensures that users feel easy and comfortable during file operations.
- Extensibility and scalability also should be considered in FTDFS. For this goal, the Java techniques will be illustrated in the design on FTDFS.
- Because of the apparent of simultaneously data passing, Java concurrency communication techniques application will be found in design process.

3.2.2 Operational Feasibility

Operational feasibility matches the requirements and expectations of users. For FTDFS, user acceptance is a significant determination of operational feasibility. It requires careful consideration of:-

- Corporate background knowledge meaning that FTDFS can run on distinct operating system, so it is apt for distinct users who are familiar with distinct operating environment.
- Effective communication means user can access files stored in other computers as soon as his/her request is sent out. Vice versa, if other computers send request to the user, user's computer responds to the request as soon as possible.
- Free tool kit is the system which is economical to be developed. Its design environment is on the basis of Java Runtime Environment which is free to download from Sun company website, so is J2SDK.
- Information is stored accurately in FTDFS however the computer is on-line or not. The users can manipulate the files operation in a friendly user interface.

- Security is the most significant factors in system advancement. FTDFS allocates username and passwords to each computer in the network. Accordingly, if users want to access the data shared in FTDFS, they must log in certain specific computer by the authorized identity.

CHAPTER 4

SYSTEM DESIGN AND ALGORITHMS

This is one of the most creative steps where functions and operations are described in detail, including screen layouts, process diagrams and other documentation. Under this chapter, several diagrams are described in detail in order to develop the basic understanding of the project.

4.1 System Design Methodology

This system is developed using various methodologies. Boehm Spiral model was selected for the advancement of the system. A key benefit of this approach is that the major issues are resolved at a very early stage. It provides the project with the versatility of a plug-in type project, meaning the ability to have core system and various add-on parts. This way it is much simpler to check various parts individually.
The Rapid Application Development methodology was chosen to implement the client-server model for DFS because it also works when the work can be broken into manageable chunks like working on various functions and classes in this project. The advantage is also that the model work fine when the project is developed in small team.

Analysis and Synthesis

The chosen methodology requires a need of constant shift between two types of design activity, analysis and synthesis. During analysis, the design was tested to check whether it is meeting goals for usability and during synthesis the design was shaped by drawing on fresh ideas born from user feedback and solutions to similar problems that have been working in the past through observation of existing systems and review of literature on existing systems.

4.2 System Architecture

The figure below is main structure diagram for our system. It follows a simple Client-server architecture. It shows how all clients are linked to each other and to the server. There can be any number of clients connected to the server and any client can upload any no. of files into the system subjected to the secondary storage at client nodes. Also multiple clients are able to download files at the same time.

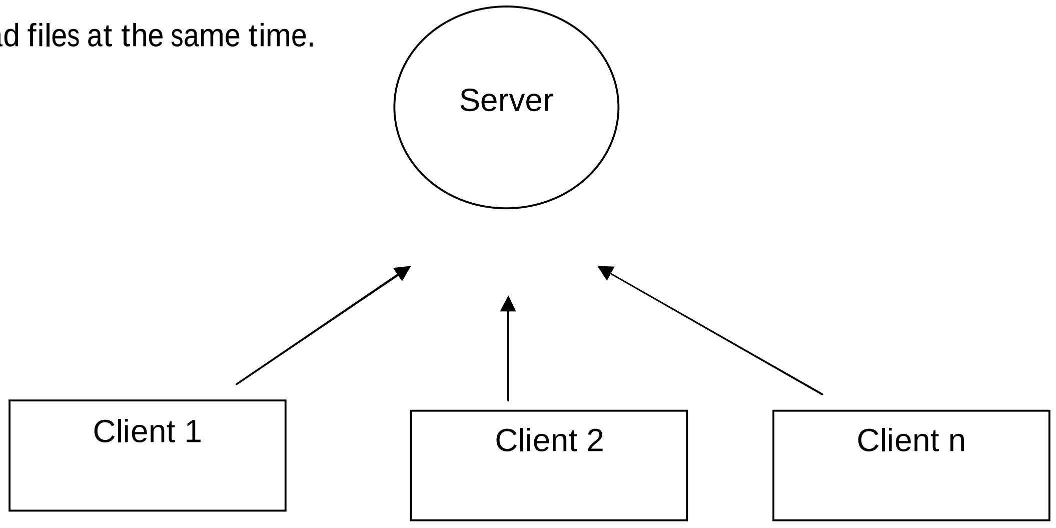


Figure 4.1: Client-server Architecture of system

4.2.1 Data Flow Diagram

Data flow diagrams are used to show graphical representation of flow of data in the system. It shows what information will be input to and output from the system, from where data will come and go and where it will be stored. A 0-level DFD for our system is shown in figure 4.2.

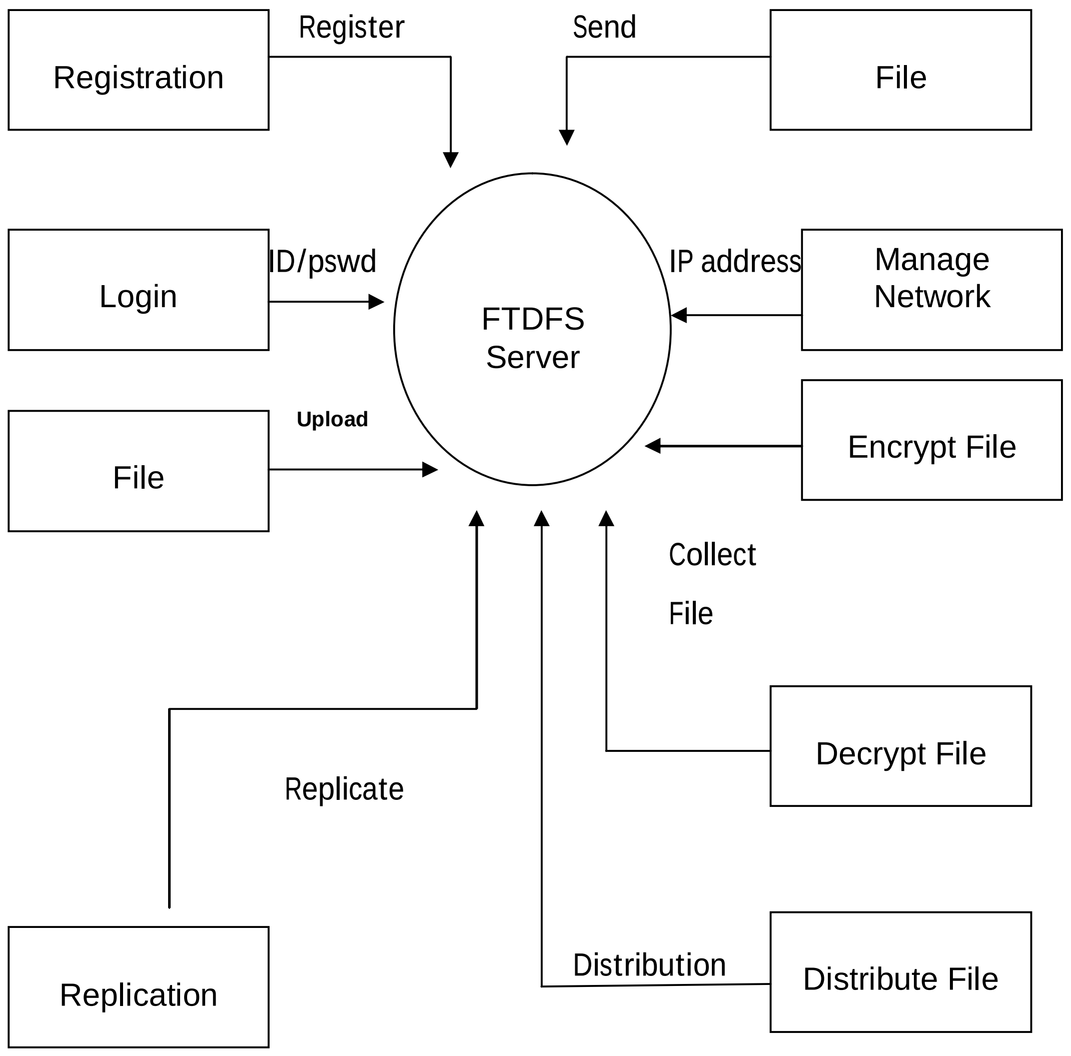


Figure 4.2: The 0-level DFD

4.2.2 Role of Client

First of all in order to login clients have to signup. After filling up all the required details, the clients get a login ID and password. After successful login, system asks client for server IP address. Client window appears after entering server IP address. Client is needed to set path where file chunks will get stored and retrieved from. Also client is required to set fault value which is decided from the values of various client properties like CPU speed, Signal Strength, Battery Life etc. on a scale of 1-10. If the fault value for client is less than a particular threshold

(5 in this case), system considers that node as faulty and it is not able to register with the server.

After successfully registering with the server, client can upload file by first choosing that file and clicking on upload Button. After successful upload server provides client with a unique key. In order to download a file, client will select the respective key and send it to server. File corresponding to that key is downloaded.

4.2.3 Role of Server

In order to access server account, one has to login with Admin authorization. After Successful Login, server window will be opened where you can start your server. As soon as any client gets registered with the server, it shows appropriate message in the message box. When a client uploads a file, server generates a unique client ID for that client from system clock. Firstly uploaded file is encrypted with the help of DES algorithm. Then it is divided into various fixed size chunks depending on the chunks size provided. These chunks are replicated and chunks and replicas are distributed among all the clients connected to the server. During the creation of chunks, a batch file is created which stores all the information about the chunks created.

Also while placing the replica it is taken care that any single node may not possess both original chunk and its replica. During the download request from client, key provided by the client is broken down to get the corresponding ID. Server now fetches the chunks from the path specified by the client corresponding to that ID. If a chunk is not found or client possessing that chunk gets disconnected, the replica is selected. When all chunks are available, server merges them with the help of batch file discussed above and we get original file but in encrypted form. Now server decrypts that file and send the original file back to the client.

4.3 Algorithms Implemented

4.3.5 Fault Tolerance Policy

FTDFS provides dual level fault tolerance. At first level it tries to remove faulty nodes/clients before they enter into the system by registering with the server. This is done by maintaining a record of system properties of all the clients like CPU speed, Signal Strength and Battery power. Their values are stored on a scale of 1-10. We have decided a threshold value of 5. If it is more than threshold, clients are able to register with the server else appropriate error message is shown. Thus in FTDFS only clients with higher reliability and availability are present. At second level, if a client possessing file chunks gets disconnected which was registered successfully in past, replication policy is used to get the lost chunk from some other client, so that file can be downloaded successfully. Thus making FTDFS a highly reliable and fault tolerant distributed file system.

4.3.2 Replication Strategy Used

In order to avoid system failures due to any client disconnection possessing file chunks, replication policy is used. After replication, replicas are placed at different client nodes. There are many replica placement policies [8] [9] [10] [11] which we studied in section 2.3 for improving quality of service and availability. Here, we have chosen a random replica placement policy because of very low overhead requirement and good performance. Many distributed systems like FARSITE [15] also use the same replica placement policy. The drawback of random replica placement policy to ignore node availability was also overcome by ensuring that only clients with higher availability are able to register to the server by keeping record of system properties of clients like CPU speed, Signal Strength and Battery power.

4.3.3 Encryption Algorithm

For the purpose of encryption/decryption, DES (Data Encryption Standard) algorithm is used. It is a symmetric key algorithm which uses 64-bit blocks, out of which 8 bits are used for parity checks (to verify the key's integrity). Each of the key's parity bits (1 every 8 bits) is used to check one of the key's octets by odd parity, i.e. each of the parity bits is adjusted to have an odd number of '1's in the octet it belongs to. It involves various substitutions, combinations and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions (for decryption). The combination of substitutions and permutations is called a **product cipher**. The main steps of this algorithm are as follows:

- Fractioning of the text into 64-bit (8 octet) blocks;
- Initial permutation of blocks;
- Breakdown of the blocks into two parts: left and right, named *L* and *R*;
- Permutation and substitution steps repeated 16 times (called **rounds**);
- Re-joining of the left and right parts then inverse initial permutation.

4.3.4 Uploading file on FTDFS

FTDFS is a distributed file system where a node in the network can upload a file on the system and file will be distributed over the network in form of chunks.

Steps for uploading a file

Figure 4.3 depicts steps for uploading a file on FTDFS which are as follows:

- In the initial step the authenticity of the user is checked. Only authenticated users are allowed to upload a file on the server. An authorized user is the one who is registered with the server.
- In the next step, a unique ID of the client is generated using the system clock (system clock ensures the uniqueness of the ID).
- The user uploads the file on the server using a secured TCP/IP connection (TCP + SSL + TCP).
- The uploaded file is now encrypted using the DES algorithm.

- The uploaded file is then divided into fixed size chunks and a batch file is maintained to store the information about these chunks.
- The chunks of the uploaded file are replicated and are distributed on the clients. The replication of chunks is done in a way such that no clients hold the replica of its own original chunks. This ensures fault tolerance within the system.
- After successful distribution of the chunks on the clients, the client is given a key file to retrieve the file.

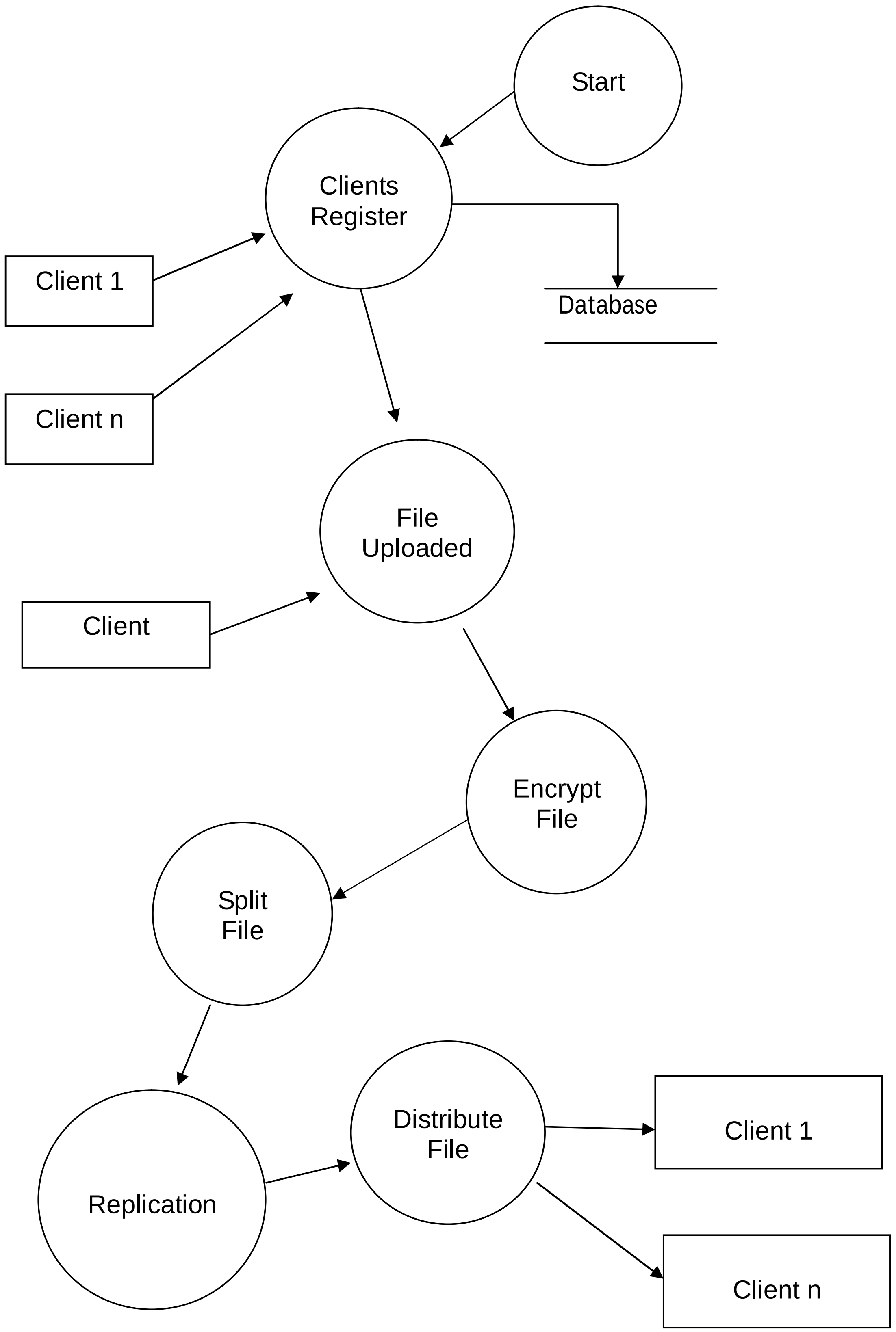


Figure 4.3: Steps for uploading file on FTDFS

4.3.5 Downloading file from System

Figure 4.4 depicts steps for downloading a file from FTDFS which are as follows:

- In order to download a file, client will select the respective key and send it to the server. The file corresponding to that key is downloaded.
- During the download request from client, the key provided by the client is broken down to get the corresponding ID.
- The server now fetches the chunks from the path specified by the client corresponding to that ID.
- If a chunk is not found or the client possessing that chunk gets disconnected, a replica is selected.
- When all chunks are available, server merges them with the help of the batch file and we get the original file but in encrypted form.
- Now the server decrypts that file and send the original file back to the client.

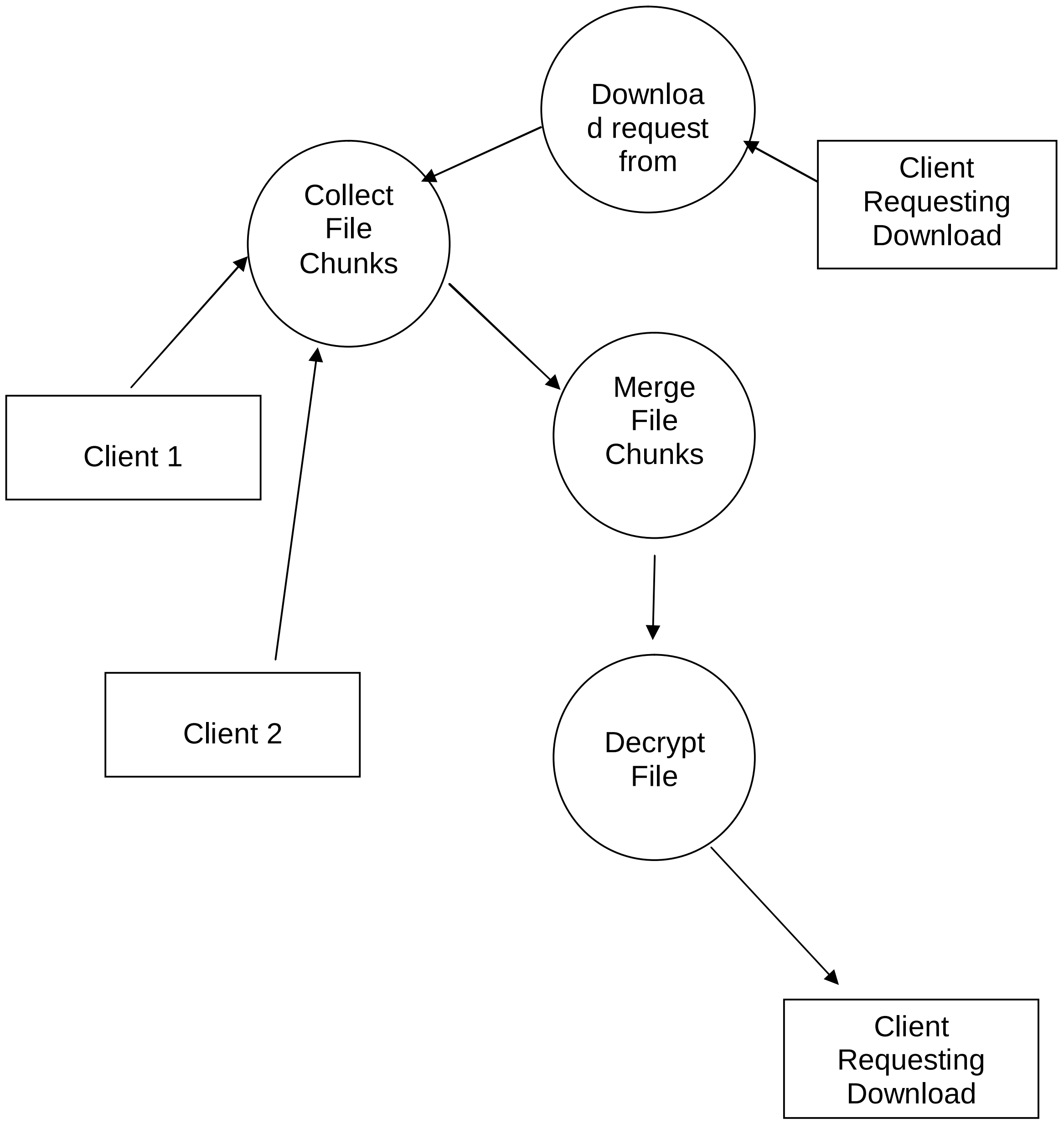


Figure 4.4: Steps for downloading file from FTDFS

4.3.6 Current Progress

Currently in FTDFS everything we discussed so far is working fine. Also right now we are trying to distribute the “key” file to all the clients in FTDFS so that any client in system can download the file.

CHAPTER 5

IMPLEMENTATION

This project has been implemented in core java and developed with NetBeans IDE 7.0.Here I have developed two main packages viz. Client and Server which contains various classes according to functions performed by client and Server.

5.1 Software Interface Prototype for Client

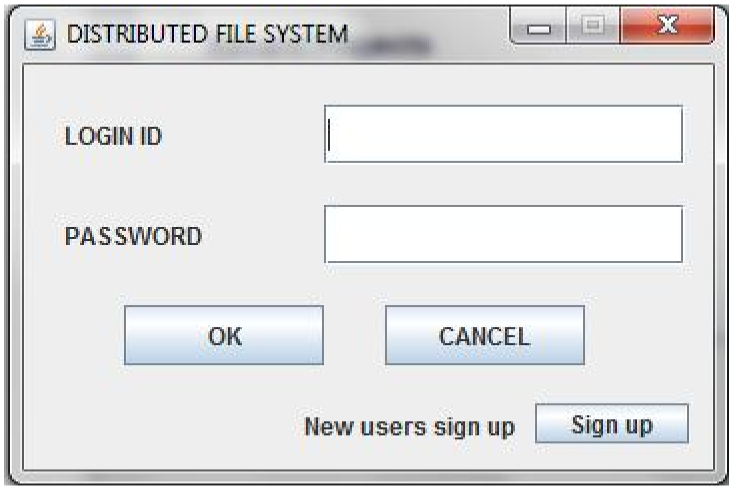
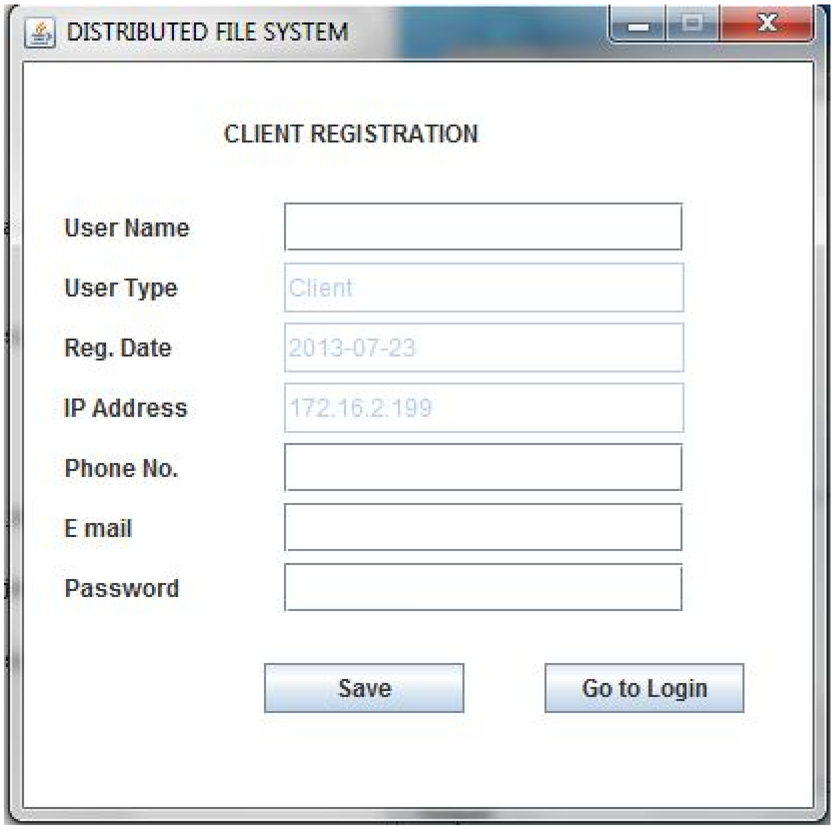


Figure 5.1: The Login Window

This is the login window for the client or the admin to log in. In order to log in with a login ID and a password, the client needs to get register first. The admin here is the server itself. When the admin logs in, it takes the admin to the server window.

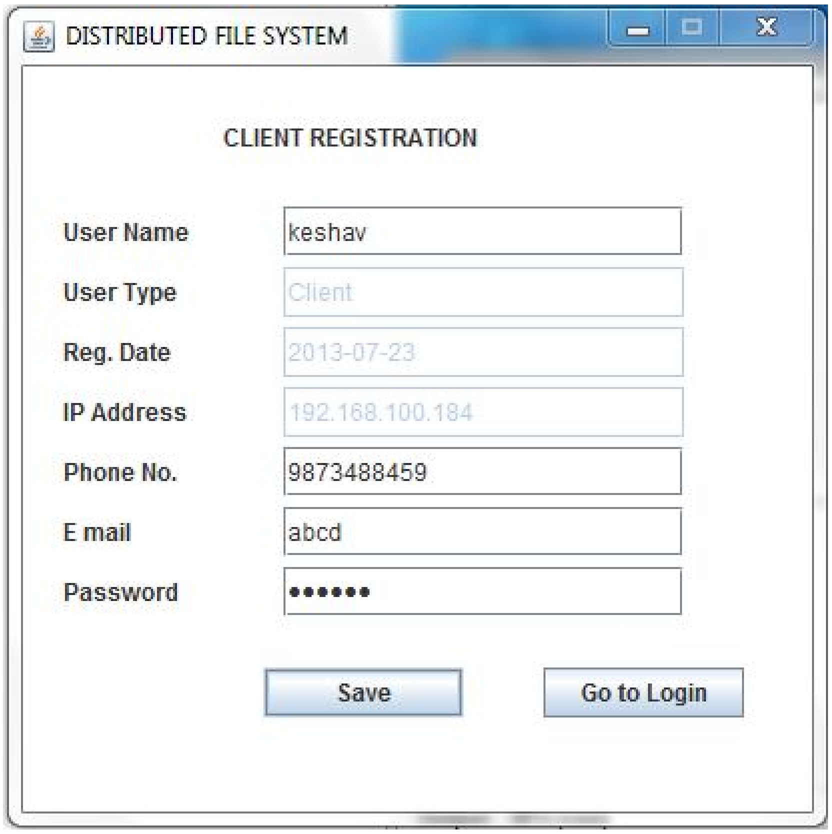

DISTRIBUTED FILE SYSTEM

CLIENT REGISTRATION

User Name	
User Type	Client
Reg. Date	2013-07-23
IP Address	172.16.2.199
Phone No.	
E mail	
Password	

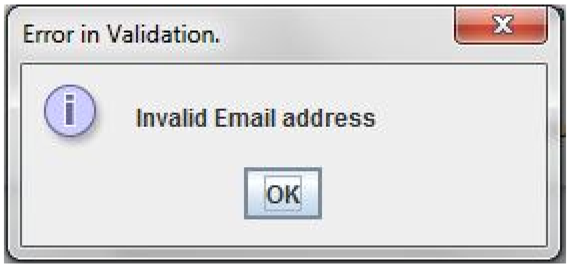
Figure 5.2: Client registration Form

This is the Client Registration window where one can register oneself. This form is fully validated. If the client misses a required field or enters something which is out of bounds or invalid, it shows an appropriate error message in a dialog box.


DISTRIBUTED FILE SYSTEM

CLIENT REGISTRATION

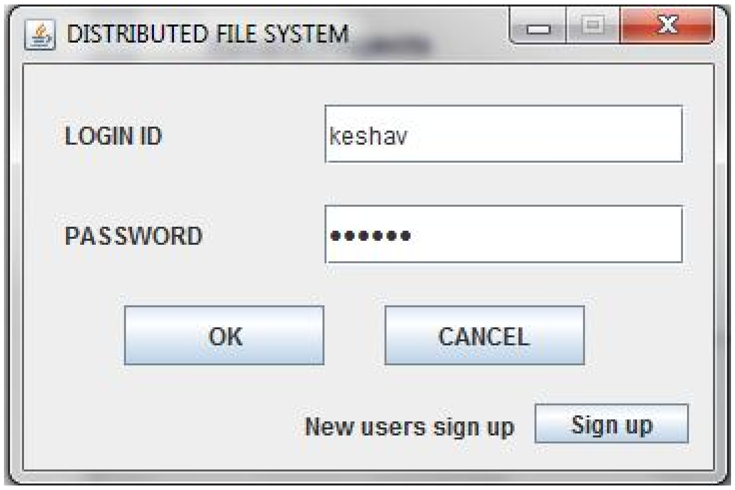
User Name	keshav
User Type	Client
Reg. Date	2013-07-23
IP Address	192.168.100.184
Phone No.	9873488459
E mail	abcd
Password	••••••


Error in Validation.

Invalid Email address

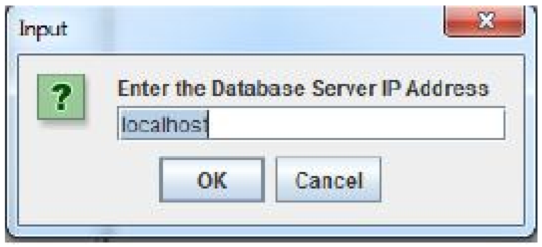
Figure 5.3: Invalid email address

Here is a small example when an invalid email address is entered during the registration process; an appropriate error message has appeared in the dialog box.


DISTRIBUTED FILE SYSTEM

LOGIN ID: keshav

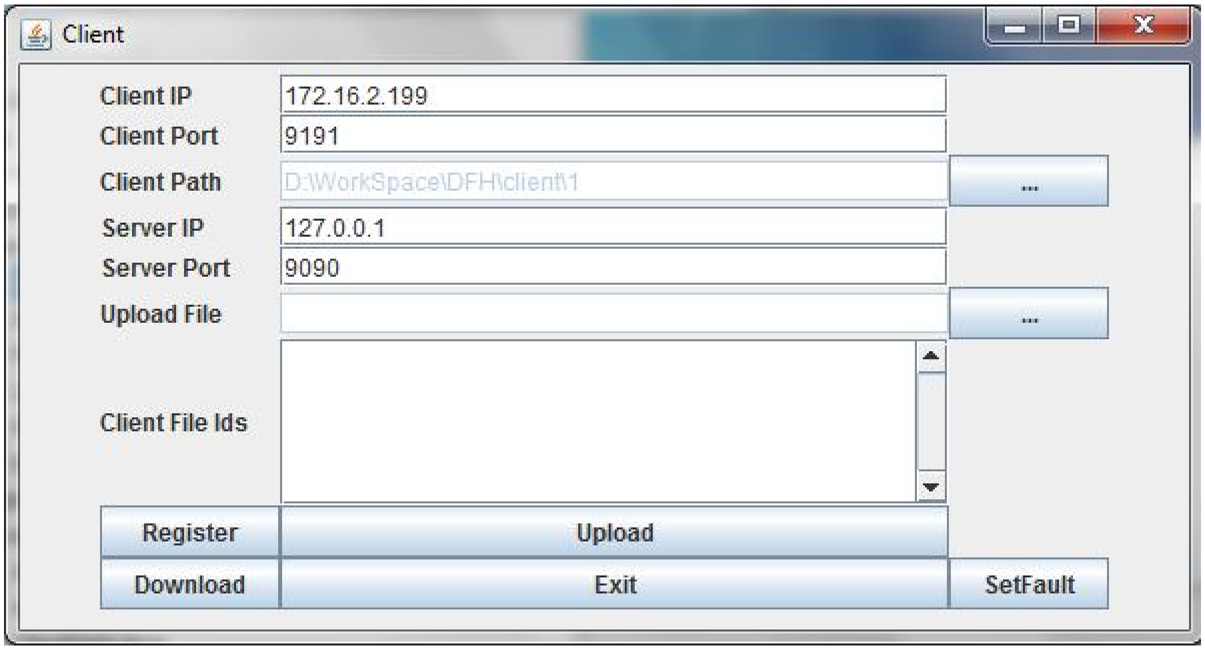
PASSWORD: ••••••


Input

Enter the Database Server IP Address: localhost

Figure 5.4: The Server IP

A client logs in, once the correct login ID and password is provided, a Dialog Box pops up asking the client to enter the IP address of the server. The client needs to enter the Server's IP address in order to get connected to the server. In this case, since the whole process in demonstrated on one system only the IP address of the server would be local host.



Client

Client IP	172.16.2.199
Client Port	9191
Client Path	D:\WorkSpace\DFH\client\1
Server IP	127.0.0.1
Server Port	9090
Upload File	
Client File Ids	

Figure 5.5: Client Window

This is the client window. The first text box is the Client's IP address which is fetched automatically and shown above. Second is the port number which should be different for all clients. Client Path is the path where the client wants to store the encrypted files in his/her system. Third is the Server IP address which the client needs to give once they log which in this case is local host. Fourth is the Upload file text field where the client gives the path of the file which they want to upload on the server. Client File Ids is the text area where the IP addresses and the port numbers of the entire client which get registered to the server is shown. This is just to let the client know that how many other clients are connected to the server at that time. Set Fault is used to set various properties of client like CPU Speed, Signal Strength and Battery Power.

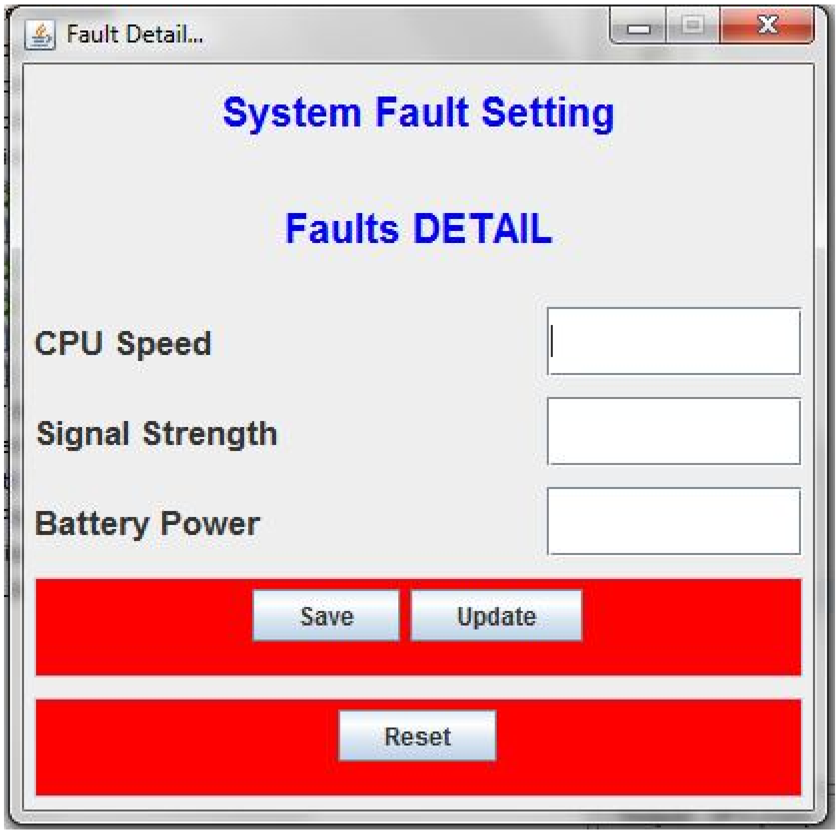
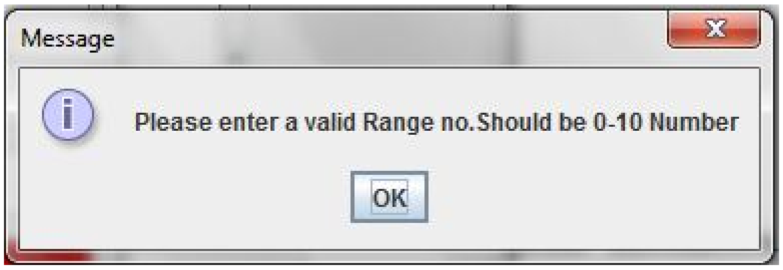


Figure 5.6: System Fault Setting

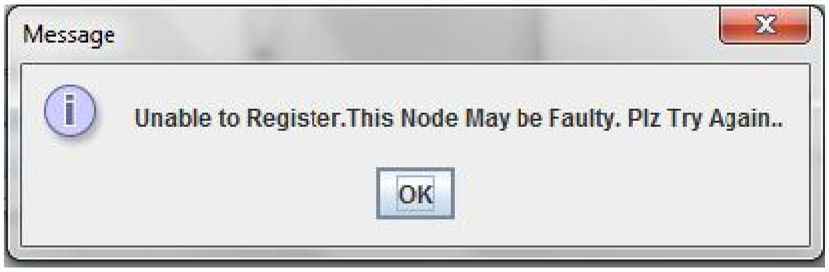
This window is to set values of various properties for the client in a scale of 0-10. If the value entered is not in range, it throws an error message.



Please enter a valid Range no.Should be 0-10 Number

Figure 5.7: Range error message

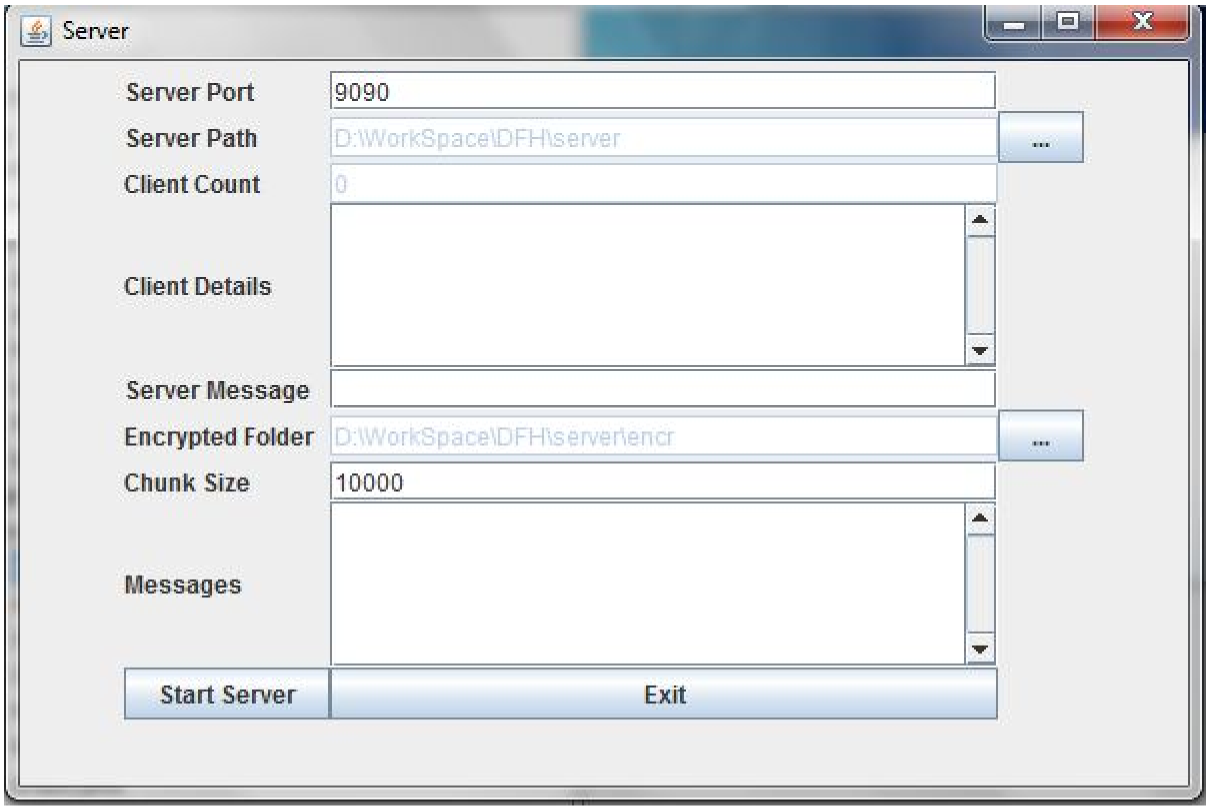
I have taken 5 as threshold value for Clients to be eligible to register to the server. The client will not be able to register if the Fault Value is less than 5. The server considers these clients as highly prone to fail and shows error message.


Message

Unable to Register.This Node May be Faulty. Plz Try Again..

Figure 5.8: Register error message

5.2 Software Interface Prototype for Server

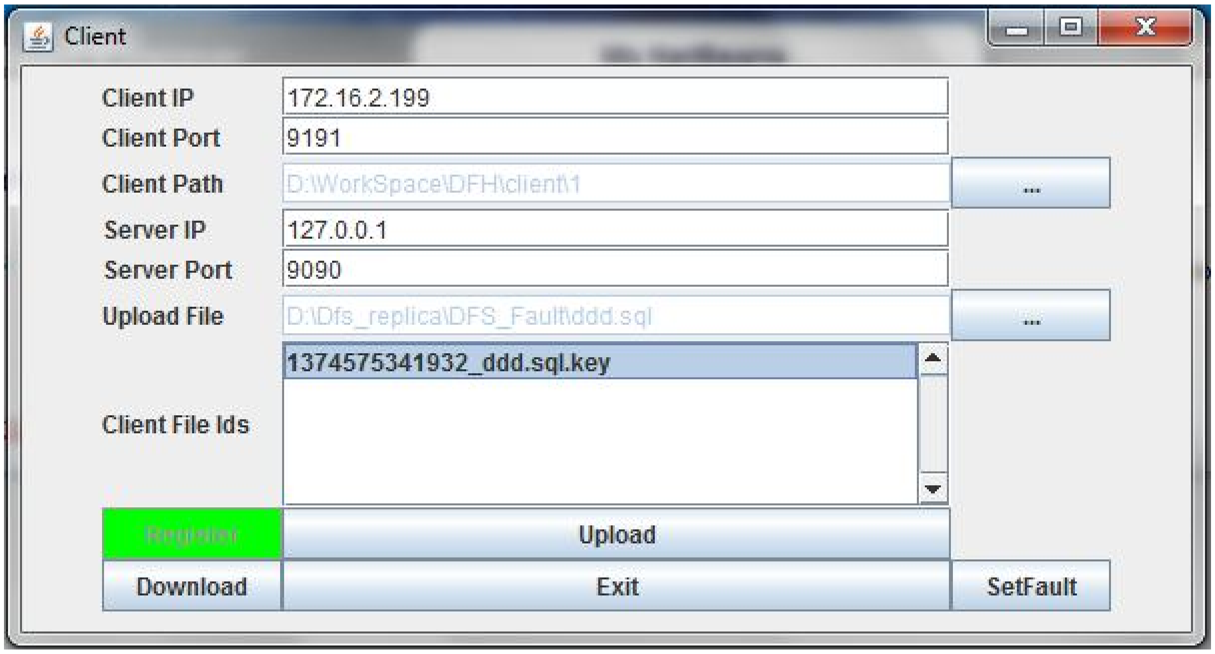

Server

Server Port 9090
Server Path D:\WorkSpace\DFH\server
Client Count 0
Client Details
Server Message
Encrypted Folder D:\WorkSpace\DFH\server\encr
Chunk Size 10000
Messages

Figure 5.9: Server Window

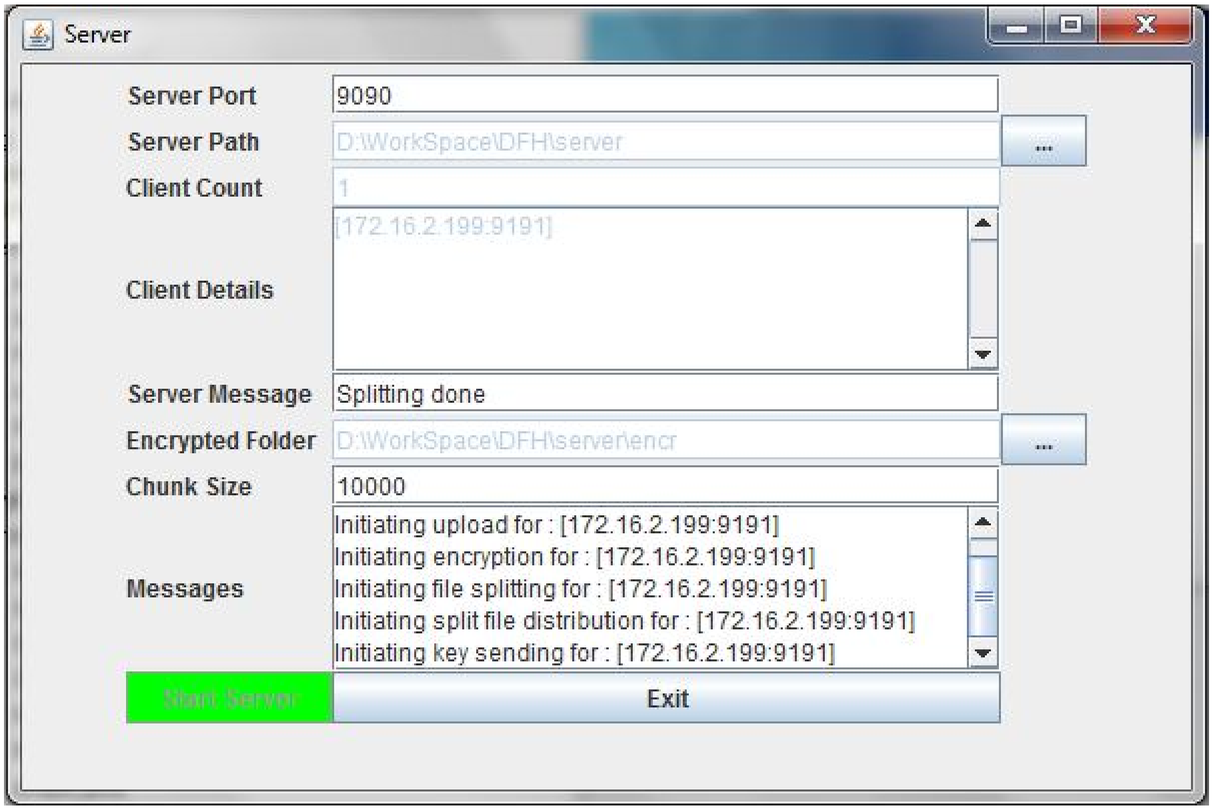
This is the Server Window, starting with the Server Port number, Server Path, Client Count which is the number of clients connected to the server. Client Details provide the clients with IP addresses and the Port numbers. Server Message shows the message such as when the button Start Server is clicked which means the server has been started. So it displays the message

Server has started. Next is the path of the Encrypted Files. Chunk Size is the size of the file in Kilobytes. In the Messages text area, the messages are displayed such as when file is uploaded on the server, it displays the message File Uploaded, when the file is encrypted, split and distributed, or when the unique key is sent to the client. All these messages are displayed in this text area.


Client

Client IP	172.16.2.199
Client Port	9191
Client Path	D:\WorkSpace\DFH\client\1
Server IP	127.0.0.1
Server Port	9090
Upload File	D:\Dfs_replica\DFS_Fault\ddd.sql
Client File Ids	1374575341932_ddd.sql.key

Figure 5.10: Client Uploading File



Server Port: 9090

Server Path: D:\WorkSpace\DFH\server

Client Count: 1

Client Details: [172.16.2.199:9191]

Server Message: Splitting done

Encrypted Folder: D:\WorkSpace\DFH\server\encr

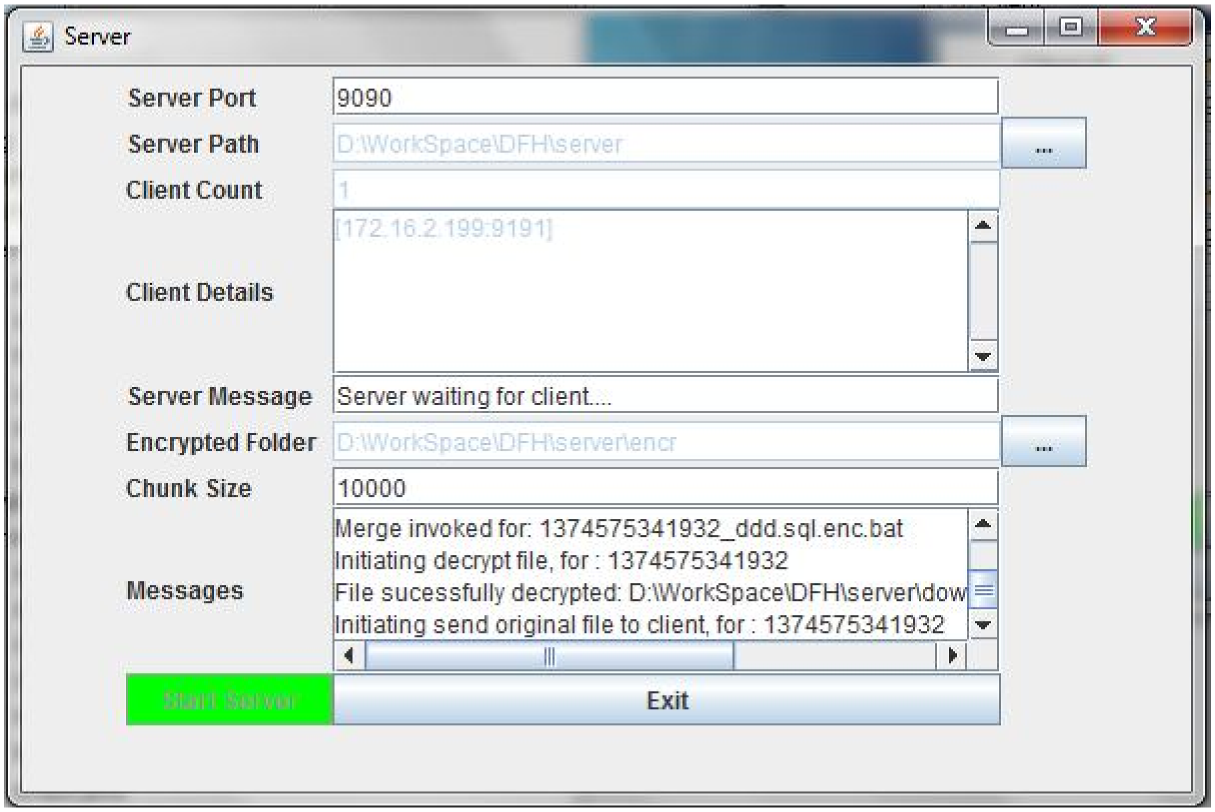
Chunk Size: 10000

Messages:

```
Initiating upload for : [172.16.2.199:9191]
Initiating encryption for : [172.16.2.199:9191]
Initiating file splitting for : [172.16.2.199:9191]
Initiating split file distribution for : [172.16.2.199:9191]
Initiating key sending for : [172.16.2.199:9191]
```

Figure 5.11: Server after File Upload

When a file is uploaded by the client, the server encrypts the file, splits and distributes it to the different clients and sends a unique key to the client who initiated the file upload.

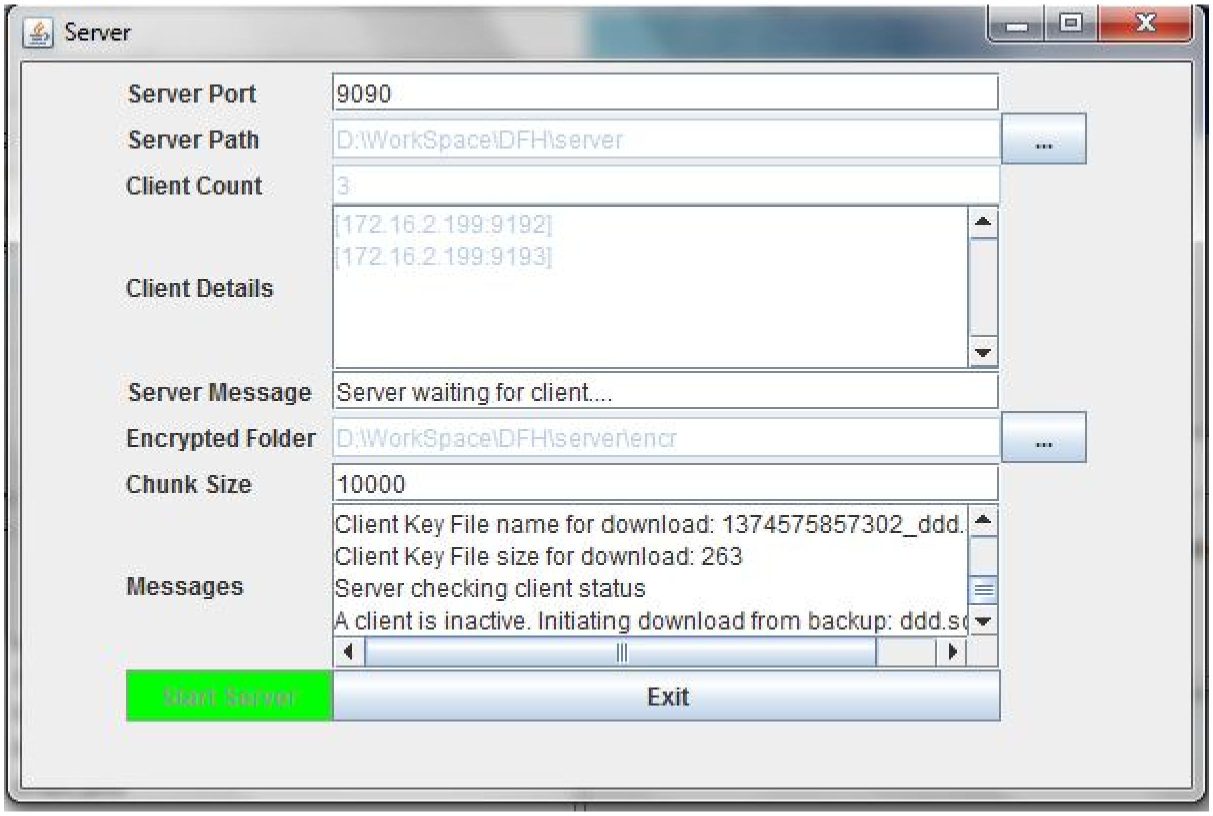


Server Port: 9090
Server Path: D:\WorkSpace\DFH\server
Client Count: 1
Client Details: [172.16.2.199:9191]
Server Message: Server waiting for client....
Encrypted Folder: D:\WorkSpace\DFH\server\encr
Chunk Size: 10000
Messages:

```
Merge invoked for: 1374575341932_ddd.sql.enc.bat
Initiating decrypt file, for : 1374575341932
File sucessfully decrypted: D:\WorkSpace\DFH\server\dow
Initiating send original file to client, for : 1374575341932
```

Figure 5.12: Server after File Download

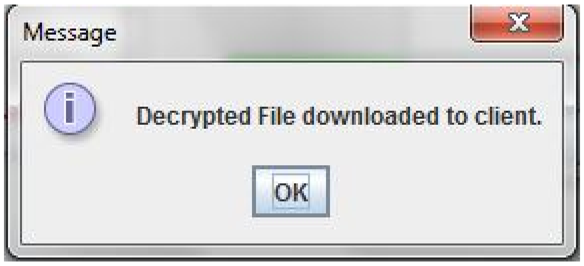
Once the key is selected the client clicks the Download button to download the file, the server merges all the chunk data from all the clients and merges them together and re-forms the original file and after decryption, sends it to the client.



Server Port: 9090
Server Path: D:\WorkSpace\DFH\server
Client Count: 3
Client Details:
```
[172.16.2.199:9192]
[172.16.2.199:9193]
```
Server Message: Server waiting for client....
Encrypted Folder: D:\WorkSpace\DFH\server\encr
Chunk Size: 10000
Messages:
```
Client Key File name for download: 1374575857302_ddd.
Client Key File size for download: 263
Server checking client status
A client is inactive. Initiating download from backup: ddd.s
```

Figure 5.13: Server after Downloading File from Replica

Also, if after uploading file a client having file chunks disconnects, the file is still retrieved from replica of those chunks.



Decrypted File downloaded to client.

Figure 5.14: File Download Message

A message is shown in a message dialog box informing the client that the file has been decrypted and downloaded to that client.

5.3 TESTING

Software testing is defined as the software product's investigation to gather information about the quality of the product. Testing is a vital process as its main concern is the verification that the software is working properly. Many attempts have been made to test every component that the system is made of. A piece of software can be tested in several ways.

The software testing methodology proposed for this project involved incremental integration testing were the application was tested as new functionality was added. Unit testing, System testing and Acceptance testing are the three phases the testing was split into during the development process.

5.3.1 Unit Testing

In the software development process Unit testing plays a vital role. A unit is smallest testable part of an application that is the classes in object orientated programming and it explores aspect of the behavior of the class.

Black-box testing and White-box testing are the two approaches to Unit testing. Black-box testing is the most commonly used since each class is represented as an encapsulated object. White-box testing is based on the method specification with each class.

5.3.1.1 Black-box Testing

Black-box testing is an approach to testing where the tests are derived from the program or component specification. The system is black box whose behavior can only be determined by studying its inputs and the related outputs; this can be also called functional testing. The observable behavior of software is examined by the Black-box testing strategy as evidenced by its outputs without reference to internal functions. The system was successfully tested for 50+ clients on single machine and with 5 clients on different machines.

Test No.	Testing Button	Expected Reslt	Actual Result	Pass/ Fail
1	Distributed File System	Login if user name and password are correct	Logged in	Pass

2	Cancel	Exits the window	Successful	Pass
3	Sign Up (Distributed File System)	Takes user to client registration form	Successful	Pass
4	Save (Distributed File System)	If no input is given in any of the required text fields, shows appropriate error message	Successful	Pass
5	Go to Login	Takes the user to Login form	Successful	Pass
6	Set Fault	Takes the user to System Fault Setting window.	Successful	Pass
7	Fault Details	If any of the column is left blank or value entered is not in range, shows appropriate error message.	Successful	Pass
8	Register(Client)	If fault value is less than threshold, shows appropriate error message.	Successful	Pass
9	Register (Client)	If no input is given in any of the required fields, shows appropriate error message	Successful	Pass
10	Register (Client)	If wrong input is given in any of the fields, shows appropriate error message	Successful	Pass

11	Upload (Client)	If file meant to be uploaded is not specified in the fields, shows the appropriate error message	Successful	Pass
12	Register (client)	If user tries to register to connect the server with the same port number, shows error message	Successful	Pass
13	Register (Client)	Once all appropriate input is given registers/connects the clients to the server	Successful	Pass
14	Register (Client)	Shows the registerd client details such as IP address and the port number in the server	Successful	Pass
15	Start Server (Server)	If no input is given in any of the required fields, shows appropriate error messages	Successful	Pass
16	Start Server (Server)	Once all appropriate input is given, starts the server	Successful	Pass
17	Upload (Client)	Once the file is selected to be uploaded, encrypts, splits and distributes the file into chunks of data	Successful	Pass

18	Upload (Client)	Sends the key to the client in order to download the file	Successful	Pass
19	Download (Client)	If the client ID key is not selected, shows error message	Successful	Pass
20	Download(Client)	If a client having chunks disconnects, download file from back up replica.	Successful	Pass
21	Download (Client)	Once the client ID key is selected, downloads the file	Successful	Pass
22	Exit (Client)	Exits the window	Successful	Pass
23	OK (Distributed File System)	If duplicate entries of username or email address are made, shows appropriate error message	Successful	Pass
24	OK (Distributed File System)	If invalid username or password is given, shows error message	Successful	Pass

Table 1: Test case Results

5.3.1.2 White-Box Testing

This type of testing is based on small portion of the program that needs to be tested. The full code will be tested this time. The results are documented and need to be studied in greater details when the testing has been completed. The outcome of such study will reveal if the software described in the requirements, behaves in a correct way. A system can be tested at several levels.

A routine can be implemented in the program to verify whether it is in working condition. This is done by creating test data.

The system integration process begins when all the components are implemented, checked and put together to form the final system. When this process is completed, the whole system is checked again and again to make sure there is no error or bug left in the software. In this way, interface problems may be discovered, as well as other types of errors.

5.4 Performance Study

Scalability: System is highly scalable as server directly fetches chunks from the path specified by the client saving lot of time. System has already been tested for high number of clients.

Reliability: System is highly reliable as if a client possessing file chunk disconnects we can still download file from the replicated chunk.

Security: File is encrypted with DES algorithm making it more secure.

Availability: Before clients are registered, it is ensured that they are less prone to faults via fault value mechanism and clients in system remains available for longer period of time.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Fault Tolerance is a very important aspect of Distributed file system. It is very necessary to handle faults in our system with proper care. We studied various fault tolerance schemes developed by the researchers over the time. Replication is one of the most popular techniques where if there is any file/chunk is not available due to any reason, its replica stored at some other location may be used for the same. In FTDFS, we are trying to minimize the occurrence of the faults. Firstly, we are ensuring that no faulty node may be able to register to the Server in the first place. We are determining whether a node is faulty or not on the basis of values of its parameters like CPU speed, Signal Strength and Battery. Also we are maintaining the replicas of the file chunks on other clients as well. So if a client possessing a file chunk gets disconnected, we are still able to retrieve the file with the help of replica present on other machine. Thus FTDFS is providing a dual level fault tolerance.

In Future, FTDFS can also be transformed into a system where out of n chunks, if some k chunks are available, the original file can be created. Also, The FTDFS can also be further developed for multiple servers in case anyone goes down the system would keep running. It might also save the space of network by certain compression technique for multiple files transmission. This System should use Data Mining techniques of AI to capture the shared files by varied permission in a huge shared file warehouse.

.

REFERENCES

1. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System,” In proceedings of the 19th ACM symposium on Operating systems principles, New York, USA,2003, pp. 29-43.
2. Tomasz Mista, “Java Distributed File System,” M. Sc Project dissertation - COMP5200, London,UK, 2011.
3. Alexandru Costan, Ciprian Dobre, Florin Pop, Catalin Leordeanu, Valentin Cristea, “A fault tolerance approach for distributed systems using monitoring based replication,” In proceedings of the 6th IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 2010.
4. Marieta Nastase, Ciprian Dobre, Florin Pop, Valentin Cristea, “Fault Tolerance using a Front-End Service for Large Scale Distributed Systems,” In proceedings of the 11th IEEE International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 2009.
5. Sunil Chakravarthy, Chittaranjan Hota, “Secure Resilient High Performance File System for Distributed Systems,” In proceedings of the IEEE International conference on Computer and Communication Technology, Allahabad, India, September 2010.
6. Swastisudha Punyatoya, “GA-Based Fault Diagnosis Algorithm for Distributed Systems”, M. Tech thesis, National Institute of Technology, Rourkela, July 2011.
7. Ioan Petri, “Quorums Systems as a Method to Enhance Collaboration for Achieving Fault Tolerance in Distributed Systems”, Informatica Economica, Academy of Economic Studies - Bucharest, Romania, February 2009, pp.68-75.
8. Bharath Balasubramanian, “A Coding-Theoretic Approach for fault tolerance in Distributed systems”, UT electronic thesis and dissertation, August 2012.
9. Bin Cai, Changsheng Xie, Guangxi Zhu, “An Effective Distributed Replication File System for Small-File and Data-Intensive Application,” In proceedings of the 2nd IEEE

International conference on Communication system software and middleware and workshops, Bangalore, India, January 2007.

10. Anna Hat, Xiaowei Jin, Jo-Han So0, “Algorithms for File Replication in a Distributed System,” In proceeding of 13 IEEE conference on local computer networks, Minnesota, October 1988.
11. Yan Chen, Randy H. Katz and John D. Kubiatowicz, “Dynamic Replica Placement for Scalable Content Delivery,” In proceedings of the First International Workshop, IPTPS Revised Papers, Cambridge, MA, USA, March 2002 , pp. 306-318.
12. Gyuwon Song, Suhyun Kimz and Daeil Seo, “Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems,” In proceedings of the First International Conference on Advances in P2P Systems, Sliema, Malta, October 2009.
13. Russel Sandberg, “The Sun Network Filesystem: Design, Implementation and Experience,” Sun Microsystems,Inc. Technical Report, (1985).
14. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, “The Hadoop Distributed File System: Architecture and Design,” In proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST), Lake Tahoe, Nevada, USA, 6 – 7 May 2010.
15. Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, Roger P. Wattenhofer, “FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment,” In proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.