ANDROID APPLICATION FOR FILE STORAGE AND RETRIEVAL OVER SECURED AND DISTRIBUTED FILE SERVERS

by

SOWMYA KUKKADAPU

B.E., OSMANIA UNIVERSITY, 2010

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Approved by:

Major Professor
Dr. Daniel A. Andresen

Abstract

Recently, the world has been trending toward the use of Smartphone. Today, almost each and every individual is using Smartphone for various purposes benefited by the availability of large number of applications. The memory on the SD (Secure Digital) memory card is going to be a constraint for the usage of the Smartphone for the user. Memory is used for storing large amounts of data, which include various files and important document.

Besides having many applications to fill the free space, we hardly have an application that manages the free memory according to the user's choice. In order to manage the free space on the SD card, we need an application to be developed. All the important files stored on the Android device cannot be retrieved if we lose the Android device.

Targeting the problem of handling the memory issues, we developed an application that can be used to provide the security to the important documents and also store unnecessary files on the distributed file servers and retrieve them back on request.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to sincerely thank **Dr. Daniel A. Andresen** for giving me a chance to do an interesting application which spreads two areas of Computer Science, Android and Distributed File Systems. I have successfully completed the Android application development only with your support and valuable ideas and suggestions throughout the development.

I want to express my extended thanks to **Dr. Mitchell L. Neilsen** for giving me his support and valuable suggestions in the development of Android application and also for providing Android devices to test the application during its development.

I would like to acknowledge **Dr. Doina Caragea** for encouraging and supporting me to develop the first Android application which helped me to develop an Android application.

Finally, I would like to thank my **Family, Friends, department Staff** who had supported me throughout the journey to and life at K-State for my Masters.

Chapter 1 - Introduction

Android is a Linux-based operating system for Android devices like Smartphone and Tablets. It is an open source platform available for users to develop Android applications using Android SDK (Software Development Kit).

The project is an Android application for storing the important documents on the secured distributed file servers. The other purpose of the application is to manage the files on SD card by storing unused files on the distributed file servers. Android platform is one of the fastest growing platforms for the mobile devices. As a result, developing an Android application is interesting to gain experience for development of Mobile applications.

1.1 Motivation

Today, each and every user uses a Smartphone for all the purposes in their daily life. Important documents are stored on the Android devices for the easy access. The chances are high of losing an Android phone. When the user loses the Android device, the user will not be interested to lose the documents stored on the Android device along with the phone. This motivated us to develop an Android application that can store the documents on the secured distributed file servers so that they can retrieve the files even if they lose the Android device.

The capacity of the hard disk for an Android device is small compared to various other high end devices like personal computers and laptops. So, the user cannot spare the memory to store the less frequently used documents on the SD card. There is a large necessity to develop an Android application that can store the files in the available memory on servers and can retrieve the files to the Android device when required.

In the Android market, there are many apps to browse files, create new files, and remove existing files on SD card. But an application to store a file securely on their space and supporting to retrieve the file later is not available for the Android device user in the market. This became the strong motivation factor for us to develop an Android application that solves the purpose of storing the file contents over the servers for later use. The following diagram is used to illustrate the overview of the application for managing the files present on the Android device.

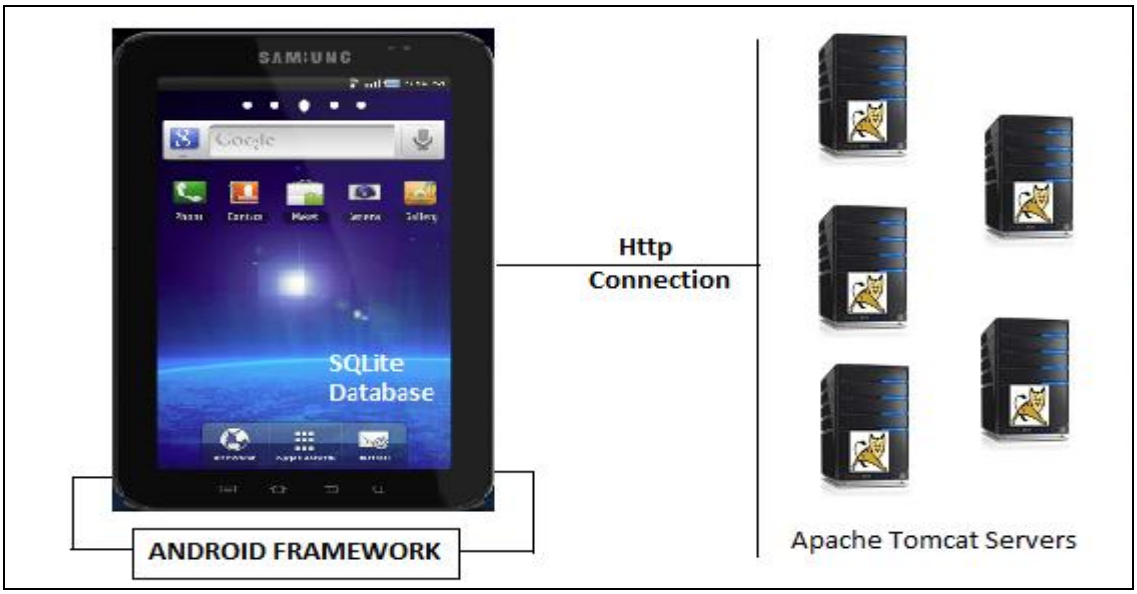

Figure 1.1- System Overview [9], [10], [12]

To store large amount of files on distributed file servers, large amount of files have to be transferred over the network. For files of large size, throughput of Computer-to-Computer peer-to-peer transfer of files is high compared to mobile-to-mobile and mobile-to-Computer file transfers. The effective throughput for the transfer of large files of mobile-to-computer file transfer is increased by dividing the file into chunks.

1.2 Project Description

The Android application developed is used to store the files on the distributed file servers securely as well as used in retrieving the stored files from the distributed file servers to the Android device. File transfer is done over the WiFi network as the speed of the file transfer over WiFi network is high compared to Bluetooth.

1.3 Applications

In the hospitals, doctors need Electronic Medical Records (EMR) to treat the patients. In order to have the knowledge about the status of the patient, he has to store EMR of the patients on his Smartphone. The capacity of the hard disk on the Smartphone does not support to store the EMRs of all the patients. This application can be used to store EMRs of the patients on distributed file servers and download the EMR of each patient to treat the patient.

Chapter 2 - Background

2.1 Android

Android is an operating system for mobile devices in the Linux environment. Open Handset Alliance developed the Android platform with the support of Google. The initial developer of the Android Inc., was bought by Google in 2005. The Androids, an open source code was released by Google under the Apache License distribution. The Android applications are developed using the hardware resources of Android devices.

2.2 Android Architecture

The components of the Android operating system are described in the following diagram.

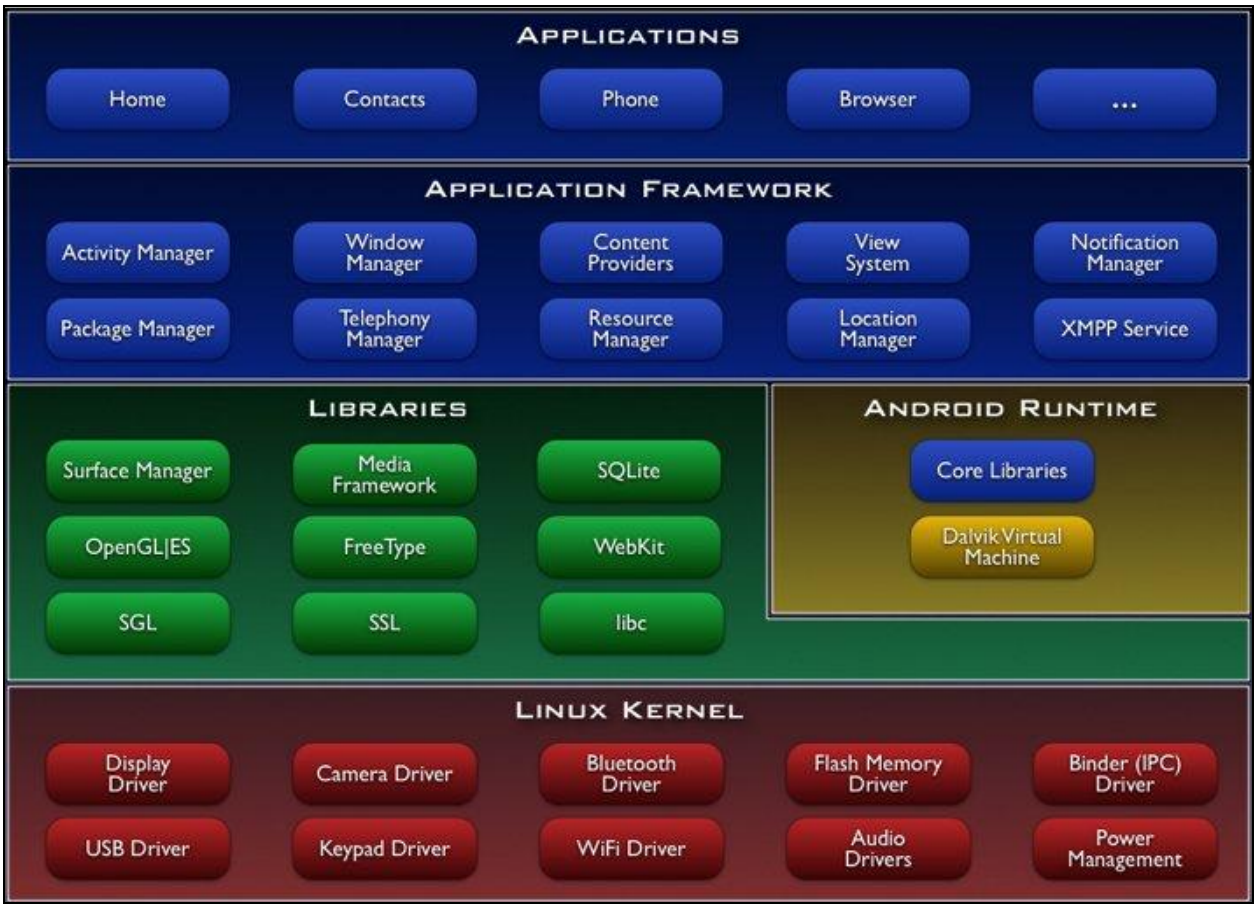

Figure 2.1- Android Architecture [1]

The Android applications are developed using the basic applications like Calendar, Camera, Maps, etc. Java programming language is used to develop the Android applications. The libraries provided by the Android are used in developing the Android application by the Android Framework.

2.3 File Storage

The different types of storing the application data in Android are:

- Shared Preferences
- Internal Storage
- External Storage
- SQLite Databases
- Network Connection

This Android application uses SQLite database to store the file information which is used later in the construction of the original file.

2.4 Security

Security has to be provided to the file content when transferred over the network and while the file content is present on the server. File content is encrypted using an encryption algorithm to provide security. AES (Advanced Encryption Standard) is used to provide security to the file content. 'crypto' package in Java language is used to create an instance of the key using 'AES' to provide security to the file-content by encrypting the file content. Decryption is done using the same key when the file is downloaded from the distributed file server to retrieve the original content on to the Android device. AES is used compared to DES (Data Encryption Standard) as it is more secured. AES uses large key size to encrypt and decrypt which makes AES a strong encryption strategy.

Chapter 3 - Requirement Analysis

3.1 Requirements Gathering

For the development of the Android application, the requirements are to be gathered to know the features that the application will support for which the application has to be tested later after it is developed. The requirements of the application are:

- Graphical User Interface to use the features of the Android application.
- Provide security to the content of the file while transferring over the network and also when it is stored on the server.
- Server to service the requests of the Android devices.
- SQLite to store the information of the files to be stored on the server used while retrieving the file later.
- Creation of the original file with the proper content.

To develop this application, we have to check the feasibility in the Android platform. In the process of checking the feasibility, many simple applications are developed to add modules to the project like security to the file content, connection with Apache Tomcat Server to transfer the content, and for the insert and select operations on SQLite database.

Further, we designed the architecture of the Android application, which included the functionalities of uploading the files on to the server and downloading the files from the servers.

3.2 Requirements Specification:

The software and hardware requirements for developing this Android application for client and server are:

3.2.1 Client Requirements

3.2.1.1 Software Requirements

Operating System:	Windows XP or higher, Android 2.0 or higher versions
Language:	Android SDK 2.3.3, Java
Database:	SQLite
Plug-in:	Android Plug-in
Tools:	Eclipse IDE
Debugger:	Dalvik Debug Monitor Server

3.2.1.2 Hardware Requirements

Device:	Android tablet
Minimum Space:	75MB
Connector:	Connecting Cable

3.2.2 Server Requirements

3.2.2.1 Software Requirements

Operating System:	Windows XP or higher versions
Server:	Apache Tomcat 6.0 or higher
Tools:	Eclipse IDE

3.2.2.2 Hardware Requirements

Minimum Free Space: 13.5 MB

Chapter 4 - System Design and Architecture

In this chapter, the application Architectural diagram, Use-case diagram, Class diagram, Sequence diagram and System diagram are used to explain the architecture of the application.

4.1 System Design

UML diagrams are used to explain the design of the system. The following are the UML diagrams that describe the system design and the action flow of the system.

- *Use Case Diagram*
- *Class Diagram*
- *Sequence Diagram*

4.1.1 Use Case Diagram

Use Case Diagram is used to represent user interaction with the system to describe the actions performed between the actor and the application to achieve the goals of the system.

The user of the system can be anyone who owns an Android device with lack of sufficient memory to store his/her file on the distributed file server. The actions of the user of the application are:

- Start the application
- Select the file to upload or download
- Upload the file to store on distributed file servers
- Download the file from the distributed file servers
- Close the application

The actions listed above are shown in the following use case diagram.

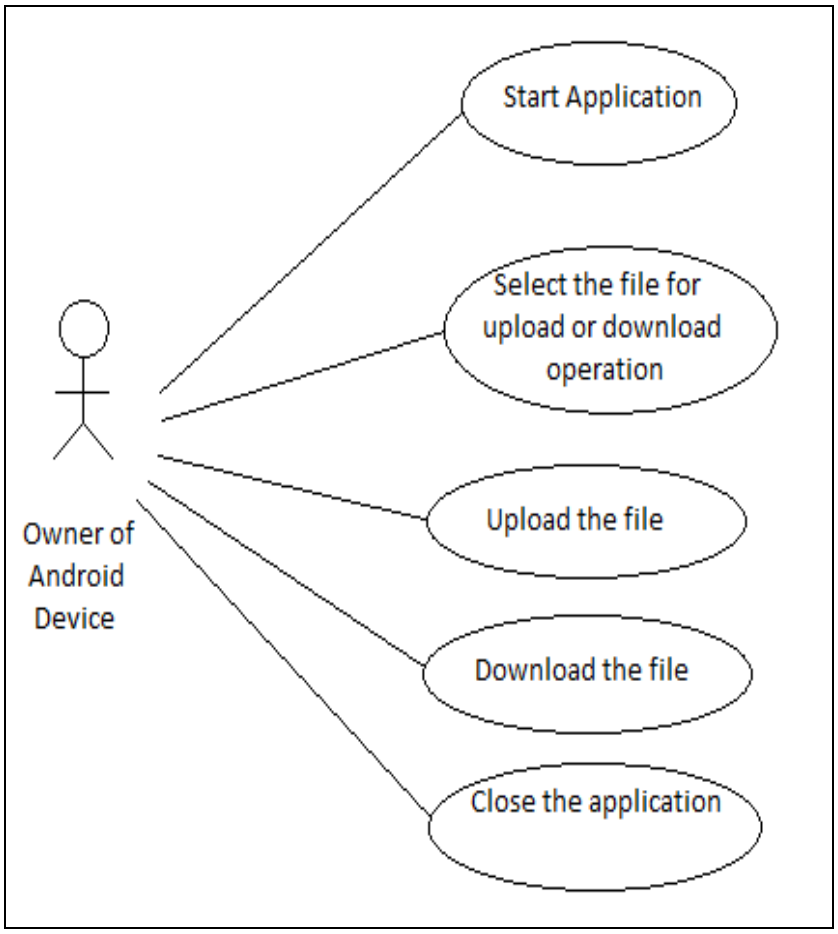

Figure 4.1 - Use Case Diagram

4.1.2 Class Diagram

Class Diagram is used to describe the structure of the system with the help of the classes, attributes, operations on the attributes and relation between the classes of the system. The classes

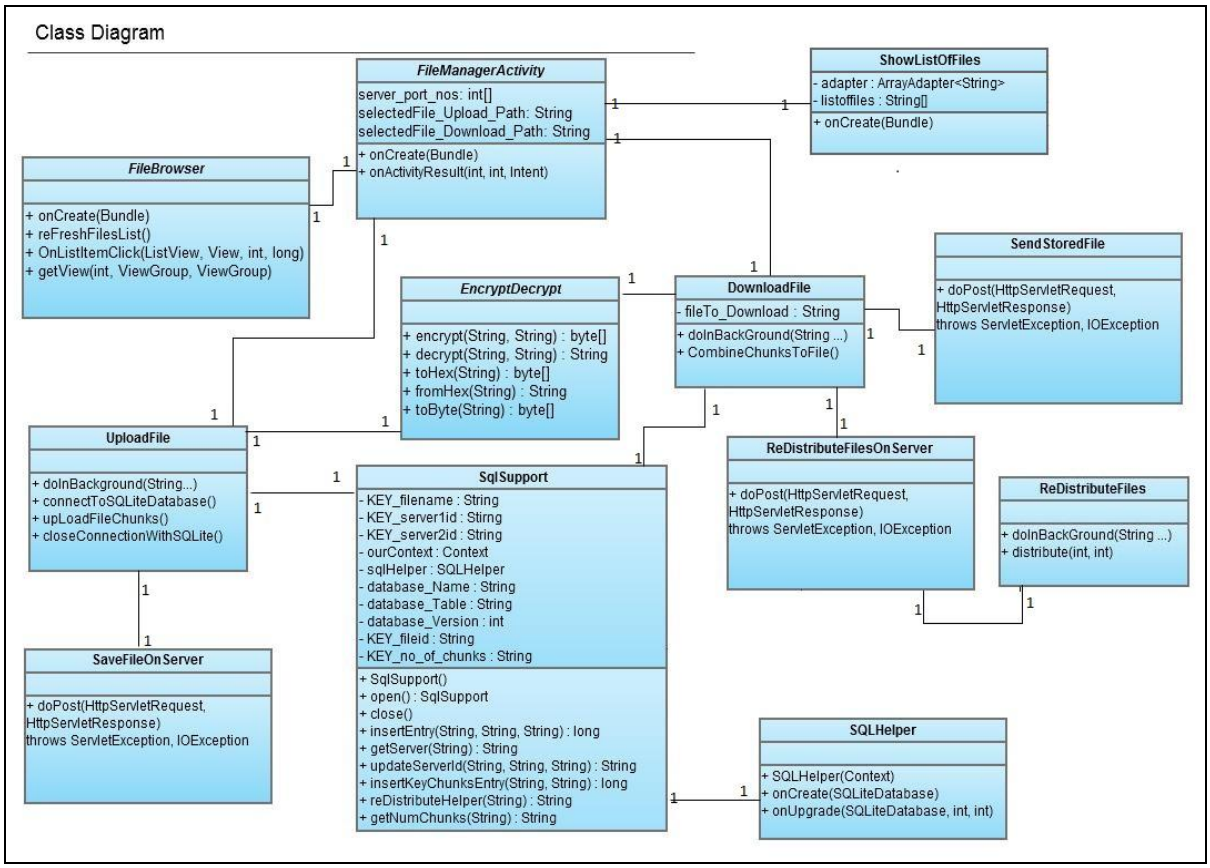

Figure 4.2 - Class Diagram

that are designed to achieve the functionality of the application are shown in the Class Diagram. It depicts the basic classes utilized to build the whole application. The functionality of each class is as follows:

***FileManagerActivity*:** It is the main activity that is used to load the application and displays the UI (User Interface) to access the features of the application. Based on the user action, corresponding activity is loaded.

***FileBrowser*:** When user selects to store the file on the distributed file servers, it is loaded to browse through the files on the SD card and sends the selected file as an input to the class which uploads the file on to the distributed file servers.

***UploadFile*:** The selected file to upload is the input to UploadFile class. In this class, the selected file is divided into chunks. Each chunk is encrypted using AES encryption algorithm. The encrypted chunk is stored on the distributed file server. Along with this, the file and server information and number of chunks are saved on the local SQLite database table of the Android device.

***EncryptDecrypt*:** It is used to provide security to the file content while it is stored on the server. It has the functionality of encrypting the file chunk before storing on the distributed file server and also, decrypting the file content retrieved from the file server to recreate the original file.

***SqlSupport*:** This class supports the methods required to store all the information regarding the files on the database as well as to retrieve and store the file information.

***SQLHelper*:** Database is created and upgraded to the new versions if any version change occurs. This is called only once when the application is installed on the Android device. And also, when any upgrades have been installed on the Android device, to upgrade the version of the SQLite database.

***SaveFileOnServer*:** After establishing the connection to the Apache Tomcat Server, this class is used to store the file content on the server space.

***DownloadFile*:** This activity is loaded when the user wants to download the file from the distributed file server using the information stored on the SQLite database. After downloading all the file chunks, the original file is constructed.

***ReDistributeFilesOnServer*:** This class is loaded when a server goes down which is determined by the timeout parameter of the http request to the server. The files stored on the inactive server are re-distributed to the other servers.

***ReDistributeFiles*:** If any of the servers goes down, the files present on that server are redistributed on to the other active servers using the duplicate files present on the other active servers.

***SendStoredFile*:** This is used to service the request from the Android device to send the files from the server to the Android device over the network.

4.1.3 Sequence Diagram

Sequence Diagram is used to show the object interactions in a sequence of time. Exchange of messages between the objects during the execution of the functionality can be depicted in the sequence diagrams. These are also used to show the processes interaction with each other in accordance of time. The following are the various sequence diagrams:

- The flow of actions performed to explain the sequence of operations for the action by the user using the Android device.
- Interaction between the classes when files present on inactive servers are redistributed among the active servers.

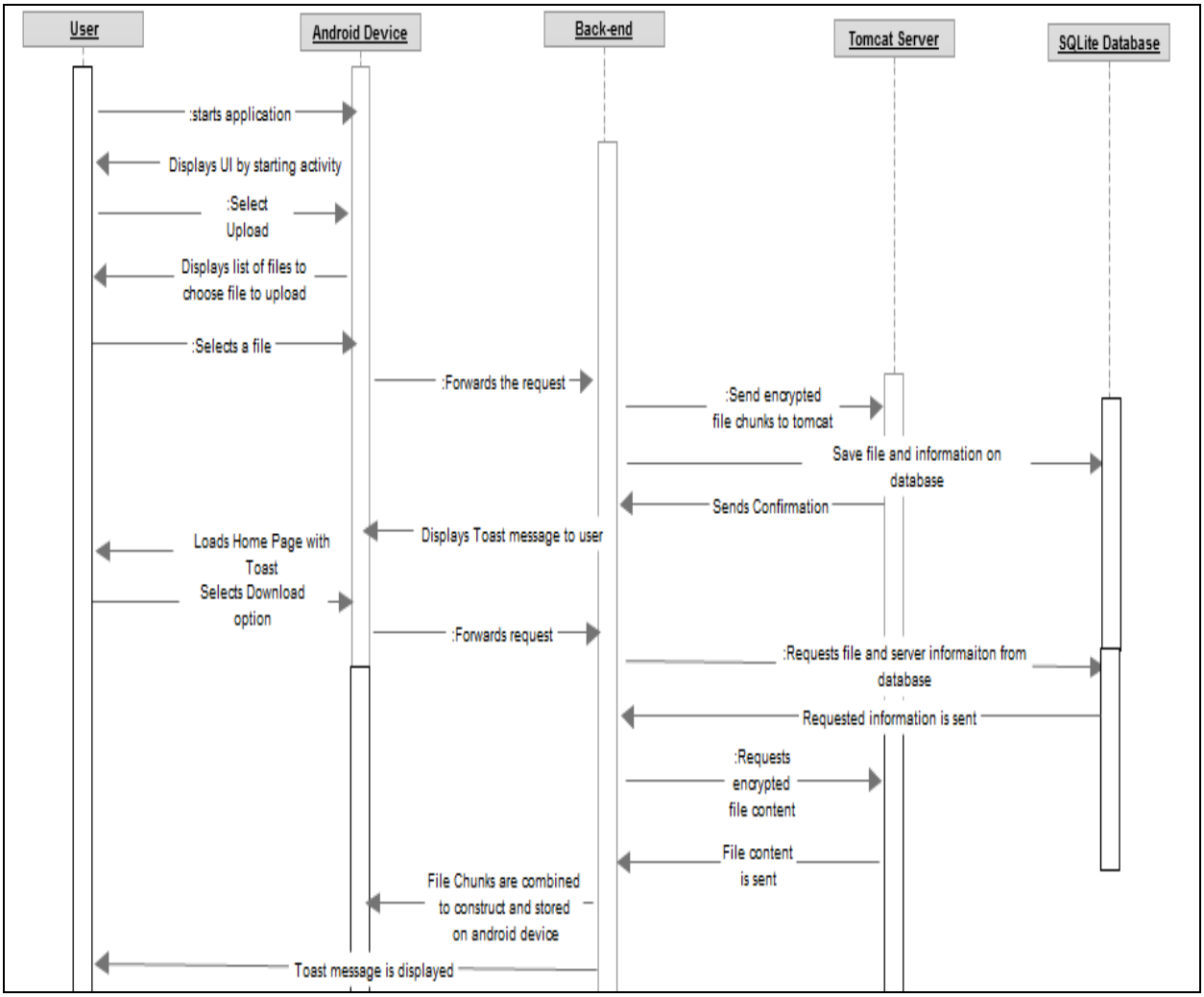

Figure 4.3 - Sequence Diagram for Features of Android Application

The above sequence diagram is used to explain the sequence of messages passed among Android device, Back-end, Tomcat Server, SQLite Database due to the actions performed by the user. The messages passed are because of the two actions performed on the user interface of the Android device. They are:

- To store the file on the distributed file server
- To retrieve the file from the distributed file server

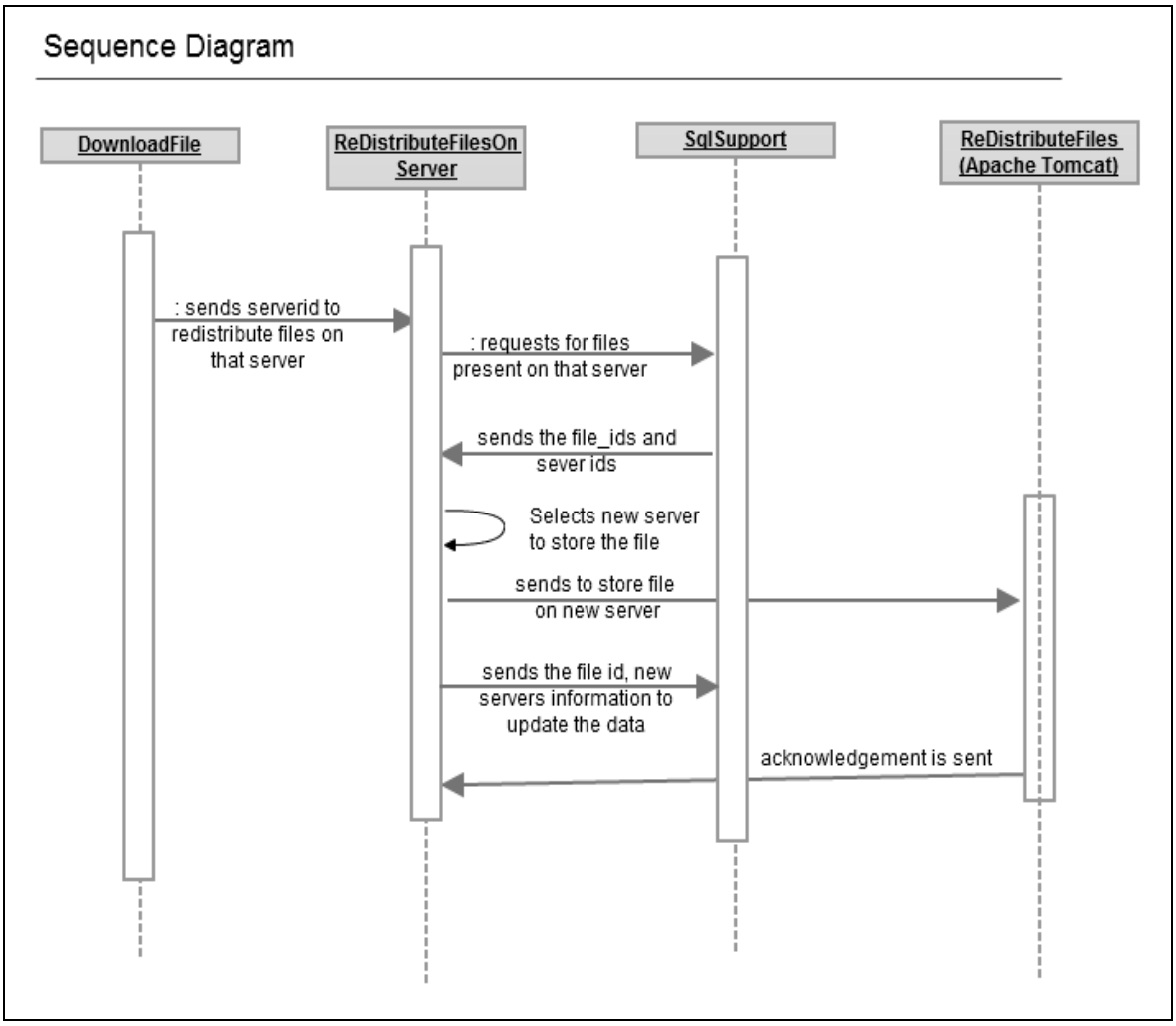

Figure 4.4 - Sequence Diagram for Re-Distribution of files on a server

The above sequence diagram is used to show the sequence of messages passed between the classes. When a particular server is down by not responding to the downloading request of the file chunk within a time limit, then the redistribution of the files takes place for the files present over the inactive server on to the active servers.

4.2 System Architecture

System Architecture of the Android application is explained in this section. As we have analyzed the design of the system with the help of UML diagrams, the architecture of the system is basically divided into two sections:

- Android Device with SQLite database working on the Android framework
- Apache Tomcat Servers

The user accesses the features of the Android application through the GUI (Graphical User Interface) on the Android device. As the user inputs requesting to access the features of the Android application, application interacts with the SQLite database. It also interacts by establishing a connection with the Apache Tomcat Server and services the request from the Android device. Apache Tomcat Server stores the files on the server memory and retrieves the files from the hard disk.

The following diagram shows the internal architecture of the system while storing the file and retrieving the file from the servers.

The steps involved in storing a file are:

- Select a file to store on the server
- Divide the file into chunks
- Encrypt the file content
- Duplicate and store them on the server

The steps involved in re-creation of the original file are:

- Retrieve the encrypted file content using one of the duplicated copies

- Decrypt the encrypted file chunk
- Combine the decrypted file chunks to create the original file.

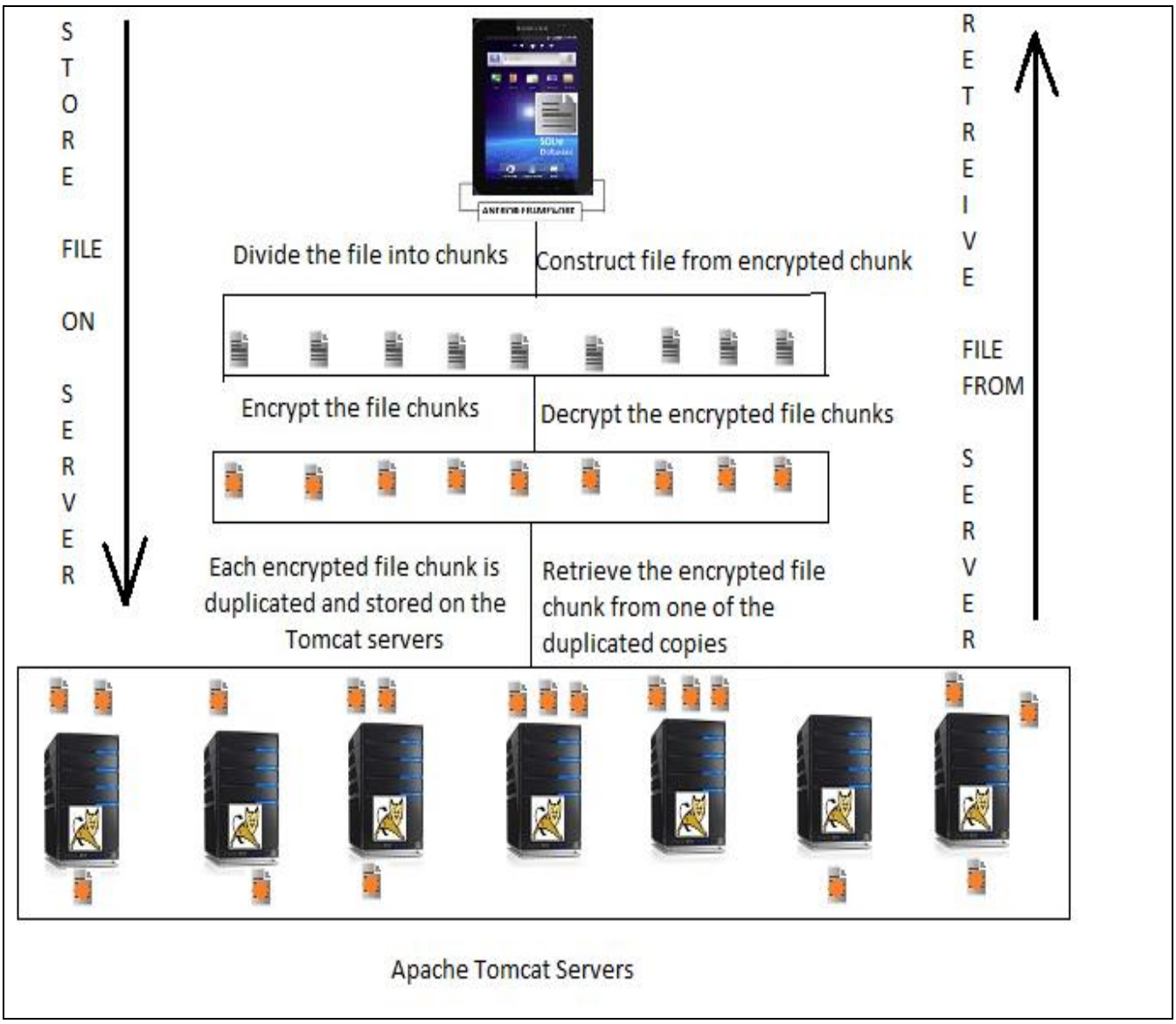

Figure 4.5 - System Flow Diagram [9], [10], [12]

Chapter 5 - Android Code Description

Android application is developed in Java using Android SDK. In this chapter the concepts that are used to build the Android application are discussed. Android has its own modules to design the application.

5.1 Android Manifest.xml

AndroidManifest.xml file is present in the root directory of each and every application. This file is used to present the Android system all the basic information required to be known to run the application code.

The description about the components like activities, services, broadcast receivers, content providers that the application uses are present in this file. The permissions to access protected API and to access other applications like access to wifi network, read-write permissions on SD card are to be specified in this file. It even specifies about the minimum Android API level required for the application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:Android="http://schemas.Android.com/apk/res/Android"
package="com.en.de"
          Android:versionCode="1" Android:versionName="1.0" >
        <uses-sdk Android:minSdkVersion="4" />
         <uses-permission
       Android:name="Android.permission.WRITE_EXTERNAL_STORAGE" />
       <uses-permission Android:name="Android.permission.INTERNET"/>
      <application Android:icon="@drawable/ic_launcher"
            Android:label="@string/app_name" >
            <activity Android:name=".FileManagerActivity"
              Android:label="@string/app_name" Android:configChanges =
             "orientation">
              <intent-filter>
                 <action Android:name="Android.intent.action.MAIN" />
                 <category Android:name="Android.intent.category.LAUNCHER" />
              </intent-filter>
            </activity>
      </application>
</manifest>
```

The minimum SDK version required for the development of this Android application to access the required resources of the Android device is 4. In order to transfer the file content over the network, "*Android.permission.INTERNET*" is specified. For the read-write operations for the file content of the Android device, "*Android.permission.WRITE_EXTERNAL_STORAGE*" permission is specified in the AndroidManifest.xml file.

5.2 Activities

Android Activity of an application is used to interact with the Android application. The features of the Android application are used by interacting with the GUI placed on the activity with setContentView(View)

Activity is initialized in the onCreate(Bundle) in which the UI components are accessed using findViewById(int) to retrieve the inputs provided by the user. According to these inputs, functionality of the system is executed.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button save_file_btn = (Button)findViewById(R.id.save_file);
      ….
}
```

5.3 Android SQLite

SQLite is the database engine on the Android platform for the Android applications. It is local for the Android device to perform database operations on the SQLite database. The database created on this database engine is only accessible to the application.

Chapter 6 - Set-Up & Implementation

An Android application is developed by creating an Android project in Eclipse INDIGO IDE in which the Android SDK plug-in is installed. The Emulator and Logcat are important tools which help in the development of Android application to test and debug the application respectively.

6.1 Set-Up

6.1.1 Tomcat Servers

Multiple servers are configured to receive the requests from the clients to store the files in the distributed environment. Multiple servers are set up by running the servers on different port numbers.

To set the tomcat server to run on a different port number, the following are the changes made:

- The server.xml file has to be edited in each folder of tomcat instance.

The following are the tags in which the changes have to be done to set the port numbers of the servers.

  - Server
  - Connector

```
<Server port="8105" shutdown="SHUTDOWN">
<Connector port="8181" protocol="HTTP/1.1" connectionTimeout="20000"
        redirectPort="8443" />
```

- The startup.bat and shutdown.bat have to be edited by directing to the startup.bat and shutdown.bat files of the main directory by setting the CATALINA_BASE and CATALINA_HOME parameters.

```
set CATALINA_BASE=C:\Users\SowmyaK\Downloads\apache-tomcat-7.0.26\tomcatinstance1
set CATALINA_HOME=C:\Users\SowmyaK\Downloads\apache-tomcat-7.0.26\apache-tomcat-7.0.26
```

The above changes are made to run the tomcat server on 8181 port number.

All the tomcat servers are made to run on different port numbers simultaneously. For each request from the client, a different thread is created to service the request from the client. HttpServletRequest and HttpServletResponse class objects of doPost method are used for communication between the client and server.

6.1.2 Android Application

The client can interact with the servers using the Android application. HTTP post requests are made for the communication between client and server. All the parameters passed from client to server are passed through BasicNameValuePair object with key and value pairs. The key of the BasicNameValuePair object is used by the server to retrieve the value of the object to the corresponding key.

6.2 Implementation

Each Android project created is associated with an activity class and is the home screen for the application. UI components of an Android application are designed in the xml files for the user interaction with the application. Each activity should be associated with an xml file to load the UI components for the user interaction.

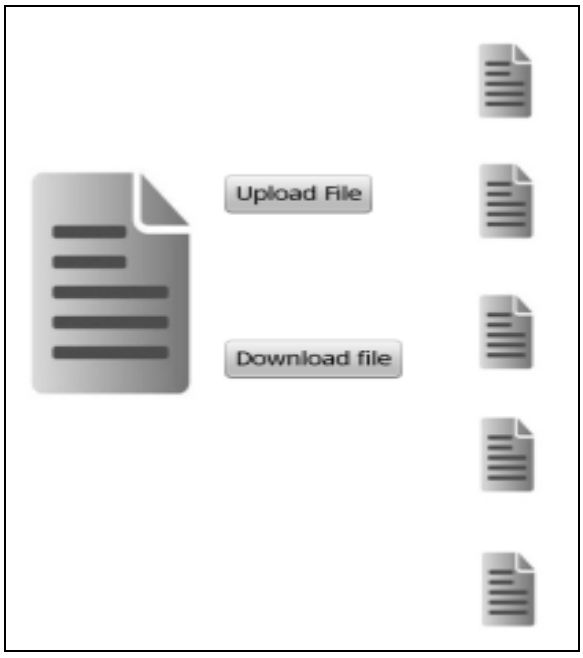

Figure 6.1 - Home Screen [11]

The FileManagerActivity is used to display the features of the main screen and select a feature to navigate the screen to a different activity with the help of the intent.

The two main features of the application are:

- Save the file on distributed file servers
- Retrieve the file from the servers

6.2.1 Save the file on Distributed file servers

The steps involved to save the file on distributed file servers are:

- Select the file
- Divide the selected file into chunks
- Encrypt the file chunks
- Select the server on which encrypted chunk has to be stored
- Send the encrypted file chunk over the network to store on selected server
- Store the file chunk on the distributed file server
- Acknowledge the client about the status of the file storage
- Based on the status the server information of a file is stored on the SQLite database

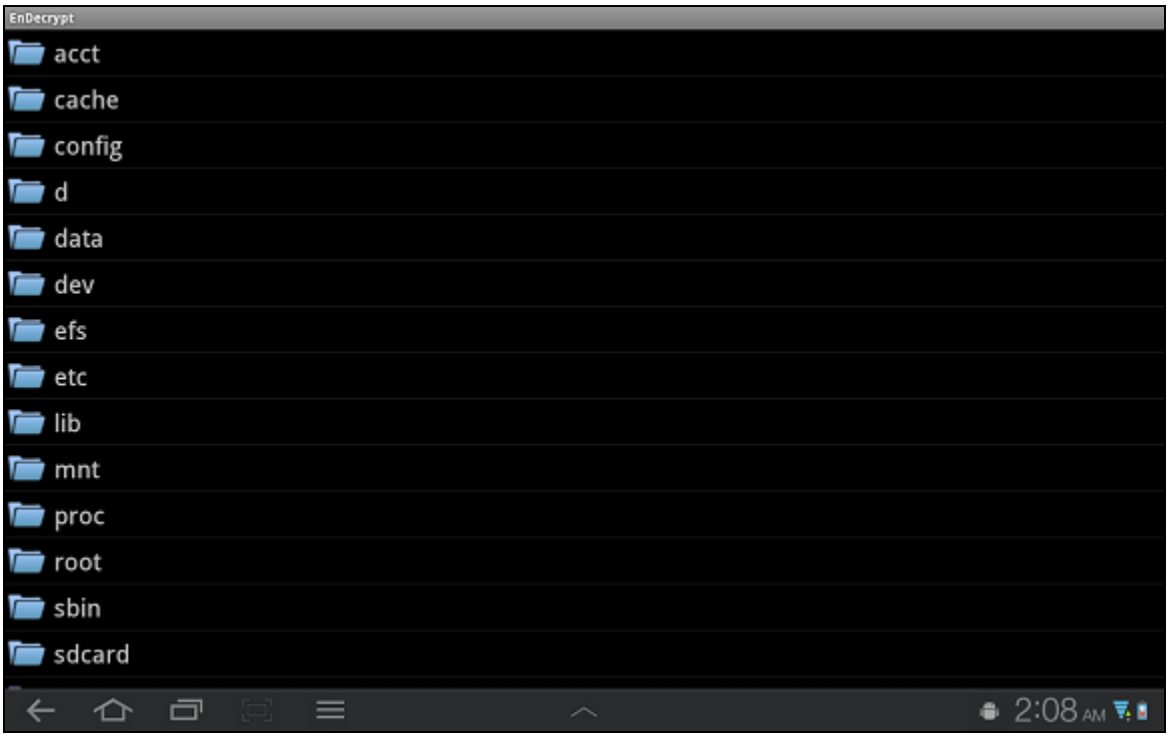

Figure 6.2 - Screen shot for Browsing the File to upload (Folder View)

Figure 6.3 - Screen shot for Browsing the File to upload (File View)

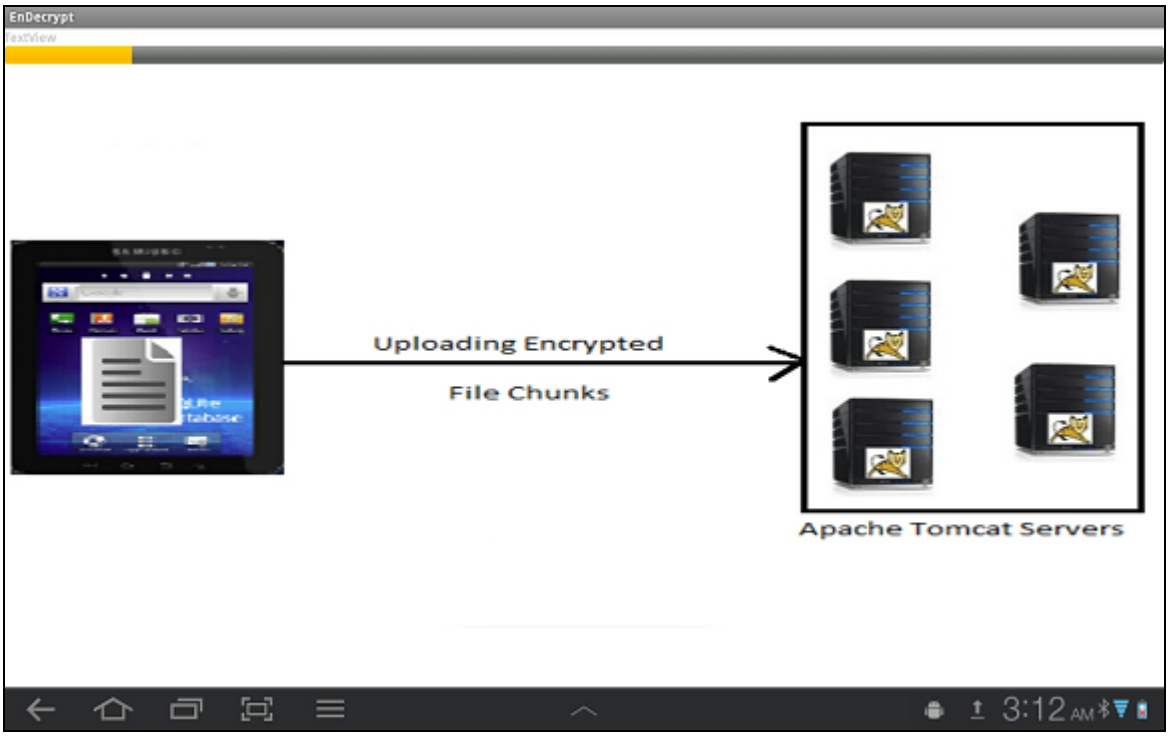

Figure 6.4 - Displaying progress of file upload on to the distributed file servers [9],[10],[11],[12]

After the successful completion of file “Toast message” is displayed on the screen and the home screen is displayed to the user.

6.2.2 Retrieve the file from the servers

When the user requires a file to be present on the Android device, then the user selects this feature.

Steps involved in building the original file on to the Android device are:

- Select the file to download on to the Android device
- Information about the file chunks are retrieved from the SQLite database
- Request the server for the file chunk
- Encrypted file chunks are retrieved
- Decryption of the file chunks is performed
- Construct the original file back from the decrypted file chunks

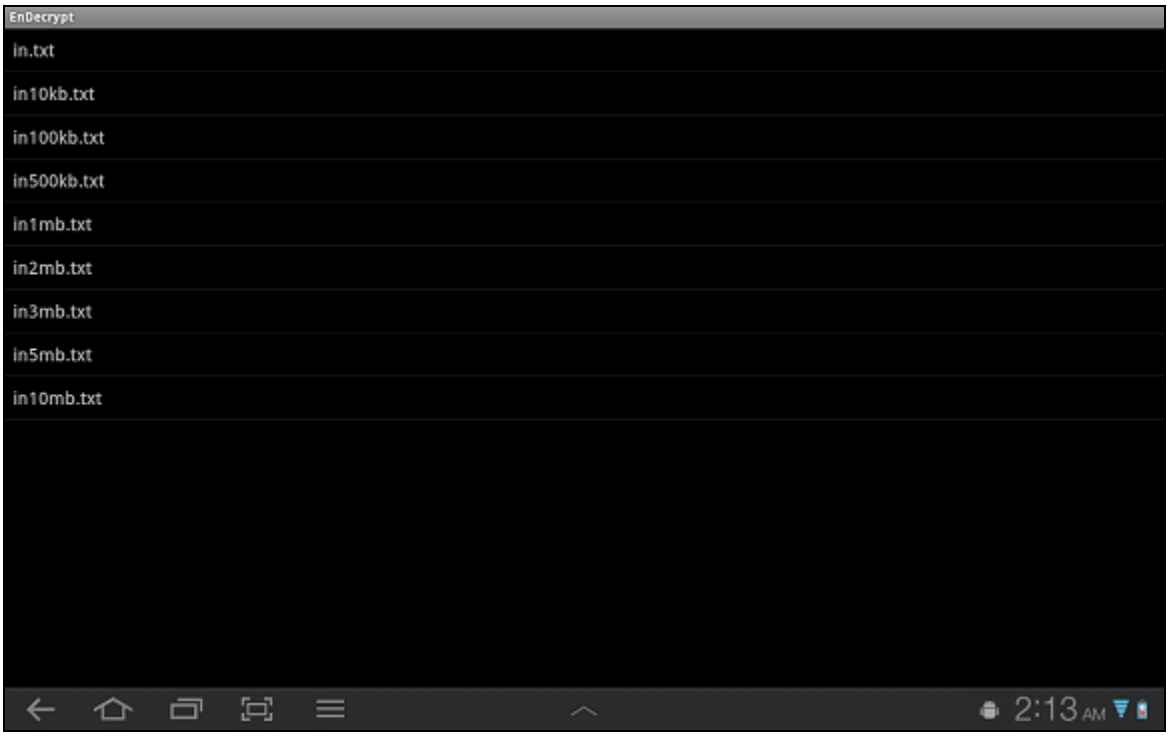

Figure 6.5 - Displays list of files stored on the servers

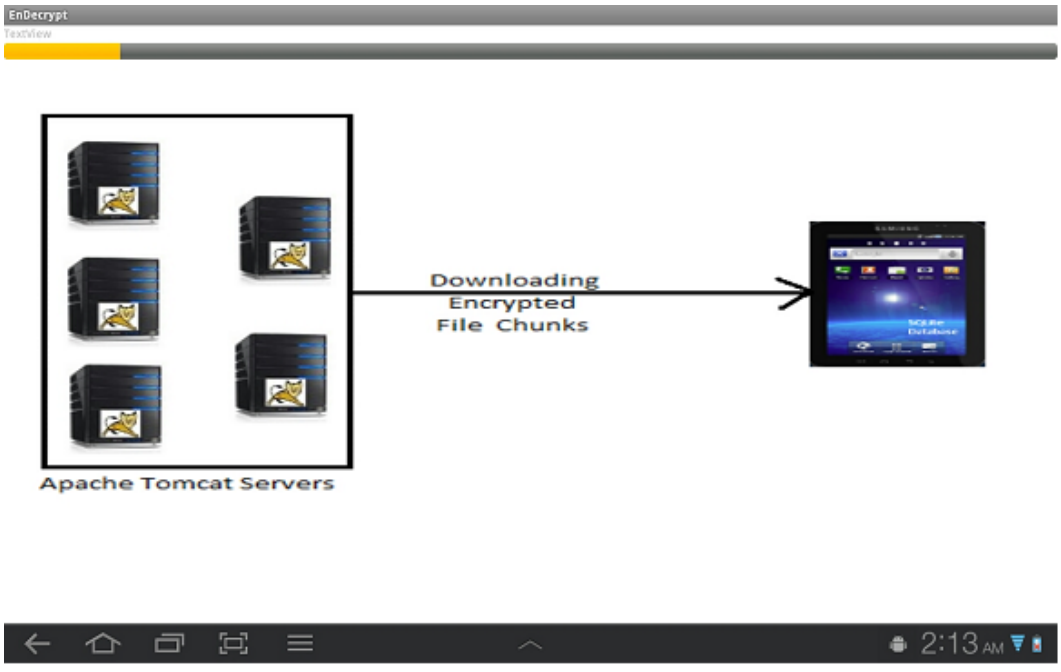

Figure 6.6 - Progress bar while downloading files [9], [10], [12]

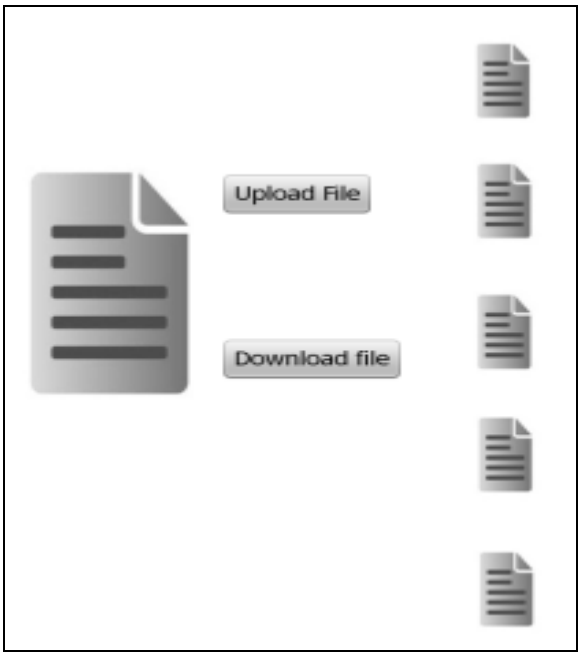

Figure 6.7 - Back to Home Screen [11]

During the retrieval of encrypted file chunks, HTTP_TIMEOUT is specified along with every request to the server. If the request is not serviced within the timeout period, then the server is down. So, in order to maintain the concept of distributed file system, we need to make an additional copy of all the files present on the inactive server. For this we perform re-distribution of files. The steps involved in the process of re-distribution are:

- Get the details of files present on the inactive server using SQLite database
- Determine a new server such that it does not have a copy of the chunk to be stored
- Send the request to the server to create a copy of file chunk on the new server
- Update the file information on the SQLite database

Re-distribution of files is done as the background process such that it does not disturb the process of original file construction. User will not be notified in the process of re-distribution as it is a background process which occurs not because of user action.

6.2.3 SQLite Database Operations

A database is created for this particular application and it cannot be shared with any other application present on the Android device. SQLiteOpenHelper is the class to be extended to create and upgrade the database.

Operations like insert, selection of columns, updating the data are performed on the SQLite database tables. All these operations are performed to store the information of files and their chunks.

The file and server information are stored in the SQLite database to use the stored information while retrieving the file from the server. "*Store_FileInformation*" is the table used to store the information like number of chunks into which file is divided and the path at which the file is present. This "*Store_Serverids*" is used to store the ids of the two servers on which file chunk is present.

The database schema of the SQLite database of Android device is as shown below.

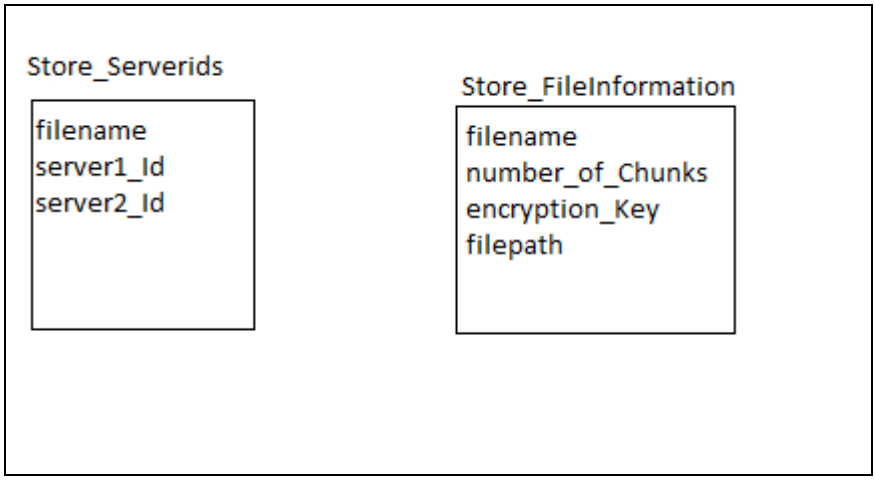

Figure 6.8 - Database Schema

The fields of the Store_Serverids table are filename, server1_Id, server2_Id. "server1_Id", "server2_Id" are the two fields used to determine the servers on which the particular file chunk, "filename", is present. This information is used in the process of downloading the file chunks from the server.

The fields of the Store_FileInformation table are filename, number_of_Chunks, encryption_Key, filepath. “number_of_Chunks” field is used to determine into how many chunks the original file is divided into, for retrieval of all the chunks to construct the original file. “encryption_Key” is used to decrypt the downloaded encrypted content from the distributed file servers. As the encryption and decryption of the value has to be done with the same key, it has been stored on the SQLite database table. “filepath” is used to re-create the file at the same place from which it is uploaded, when downloaded from the server.

6.2.4 Managing Orientation Change

The activity re-loads when the orientation of the Android device changes. In order to prevent the re-loading of the activity with the orientation change, a modification is made to the Manifest.xml file.

Android:configChanges = “orientation” has to be added to the activity tag in the AndroidManifest.xml file.

```
<activity Android:name=".FileManagerActivity"
        Android:label="@string/app_name" Android:configChanges =
        “orientation”>
</activity>
```

The details of the number of lines of code for the implementation of this Android application are as follows:

Category	Number of lines of code
Java	1069
XML	85

Table 6.1- Number of Lines of code

Chapter 7 - Testing

In developing the Android application, the Dalvik Debug Monitor Server (DDMS) is useful for the errors that have occurred by displaying the log messages in the Logcat. It comes along with Android SDK plug-in during the set-up for the Android application development environment on Eclipse IDE.

The different testing strategies used to test the Android application during the development of Android application are discussed in this chapter.

7.1 Unit Testing

In this testing, each unit is tested independent of other units during the development of the Android application. For this testing, I have treated the functionality of the system as a unit and tested each unit. The following are the unit test cases considered for performing unit testing.

Class Name	***Test Case***	***Expected Result***	***Observed Result***	***Result***
FileManagerActivity	Alignment of components	Proper alignment	Aligned properly	Pass
UploadFile	Divide file into chunks	Each chunk of 1MB size	All chunks of 1MB size	Pass
UploadFile	Encrypt file content	Proper encryption	Encrypted correctly	Pass
UploadFile, DownloadFile, ReDistributeFiles	Connection with SQLite database	Obtain connection with SQLite database	Obtained connection	Pass
SqlSupport	Operations on SQLite tables	Create and operations with data on SQLite database tables	SQLite database tables are created with operations on data	Pass
DownloadFile	Decrypt the file content	Correct decryption	Decrypted correctly	Pass
DownloadFile	Combine files to original file	Construction of original file	Original file is built with its contents	Pass
SaveFileOnServer	Create a new file on server and store its contents	File creation along with its contents	Files is created with its contents	Pass
SendStoredFile	Read the file	Read the files	File contents are	Pass

	content of a file stored on the server and send to client	properly and send back the file content	read properly and sent to the Android device	
FileBrowser	Display of all the folders and files	To display all sub-files and sub-folders present when an item is selected	Listed all the sub-folders and sub-files in each directory	Pass

Table 7.1 - Unit Testing Results

7.2 Integration Testing

Integration testing is performed when all the tested unit cases are integrated to for the proper full functionality of the system. Message passing between the units are tested for the format and desired content.

Class	***Test Case***	***Expected Result***	***Observed Result***	***Result***
Navigation	Action Call	Corresponding action is called	Corresponding action is invoked	Pass
Communication	Between Activities	Parameter passing and result of the action	Parameters and results are retrieved properly	Pass
Communication	Between Android device and server	Connection establishment and Parameter passing	Established the connection with the server and message is retrieved	Pass
Communication	Between Android device and server	Results for the corresponding action	Acknowledgement and messages are passed corresponding to action	Pass
Communication	Between units	Messages passing as arguments and results	Format of messages passed and results of the functionality	Pass

Table 7.2 - Integration Testing

7.3 Performance Testing

7.3.1 Number of Users

In order to test the performance of the application with respect to Apache tomcat server running on the system with the following specifications, I am using Jakarta-Jmeter tool to perform testing.

System Configuration:

Operating System: Windows 7

Processor: Intel® Core™ i3 CPU M370 @ 2.39GHz

RAM: 4GB

Test-case details:

Number of threads: 500

Loop count: 2

Total simulated users accessing the server: 1000

This testing is used to determine the performance of the server handling the requests for the file-write and file-read operations of the file content sent from the Android device. The file-read and file-write operations are performed on the hard-disk of the server. The performance is determined for these operations handled when 1000 users are sending the requests at the same time to the server running on the above mentioned system configuration.

Parameters used to determine the performance for the test case in handling the requests sent from the Android device are Throughput and Average Response time.

From the following graph, it is found that the server can handle 41958 requests per minute and the average response time is 32ms for responding back to the Android device when 1000 users are sending requests of type file-read and file-write operations to access the server simultaneously.

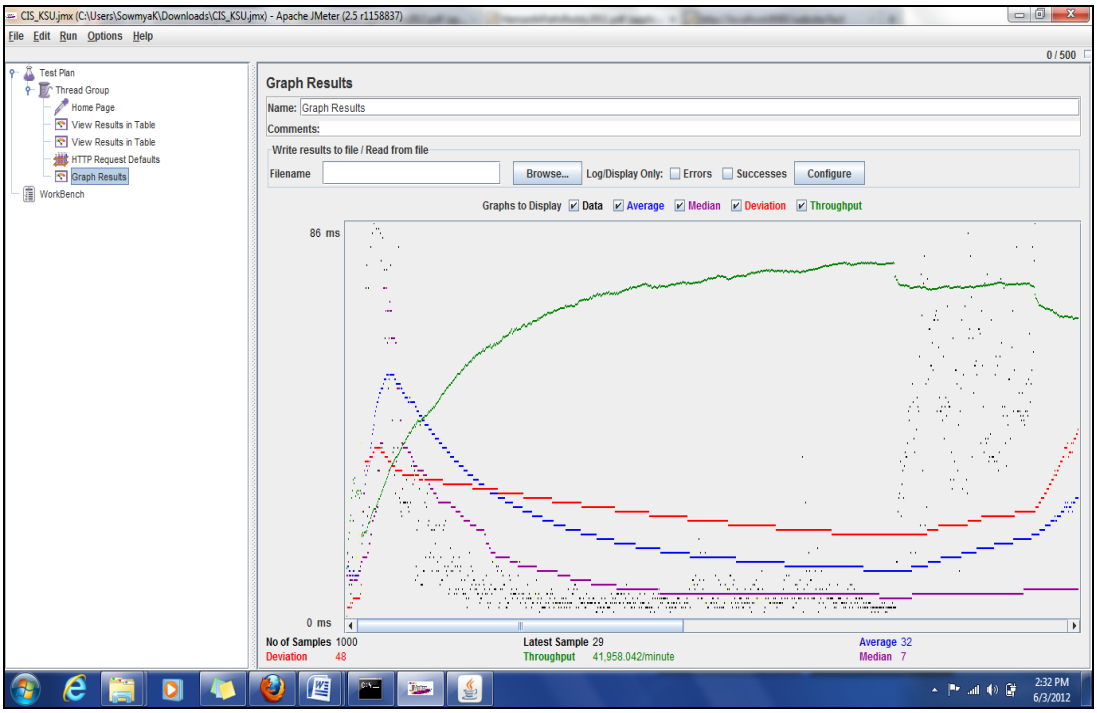

Figure 7.1 - Performance Testing Results (JMeter) and Server running on same Machine

For each request, the server has to service a memory read or memory write operation on the hard disk. For storing the file on the server, we have a memory write operation and for sending back the stored file content, it is a memory read operation. In order to access the disk, the server spends lot of time for the write and read operations. So, the server is able to handle 41958 requests per minute and the average response time by servicing the request from the Android device is 32ms.

If the server and the jMeter to test the performance of the application are run on the different machines the results of the performance of the android application vary. The following is the screen shot of the jMeter tool when the jMeter is run on the following System configuration.

Operating System: Window 7

Processor: Intel® Core™2 CPR 6600 @ 2.40 GHz

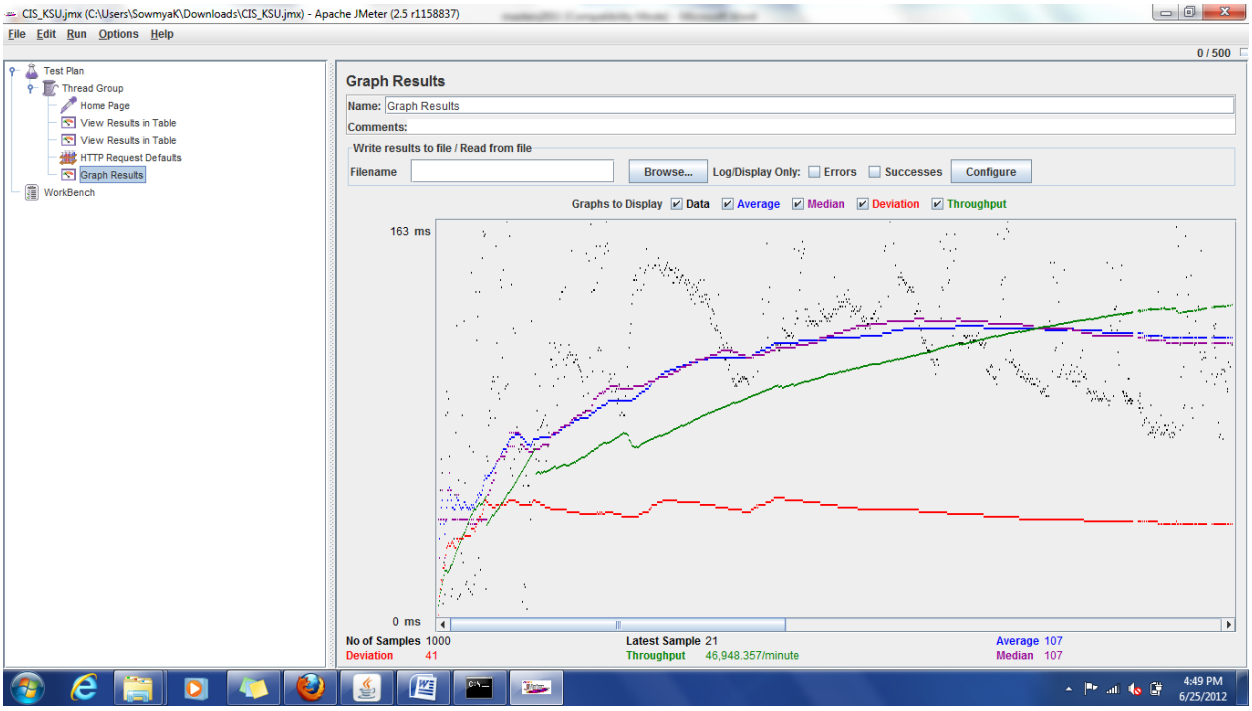

Figure 7.2 - Performance Testing Results (JMeter) and Server running on different Machine

By comparing the results from the Figure 7.1 and Figure 7.2, the performance of the android application can be increased if the RAM of the Server can be increased.

7.3.2 Number of Servers

As the Android application sends request to multiple servers at a time for the creation of duplicated file chunks, the performance of the system is measured with respect to the number of servers.

The difference in the time taken to transfer the files on to the server by varying the number of servers is observed significantly only for a file of size at least 1MB. This significant identification is observed only for 1 MB sized file because of the chunk size which is considered as 100Kb.

As the number of servers is increasing, the time taken to complete the task of storing the file on the distributed file servers is reducing. This is because of the distribution of the work of storing the file chunks among the increased number of servers. Thus, the application is able to complete the task fast on more number of servers compared to the less number of servers.

As shown in the Figure 7.2, the bandwidth of the file transfer is increasing as the number of servers increase.

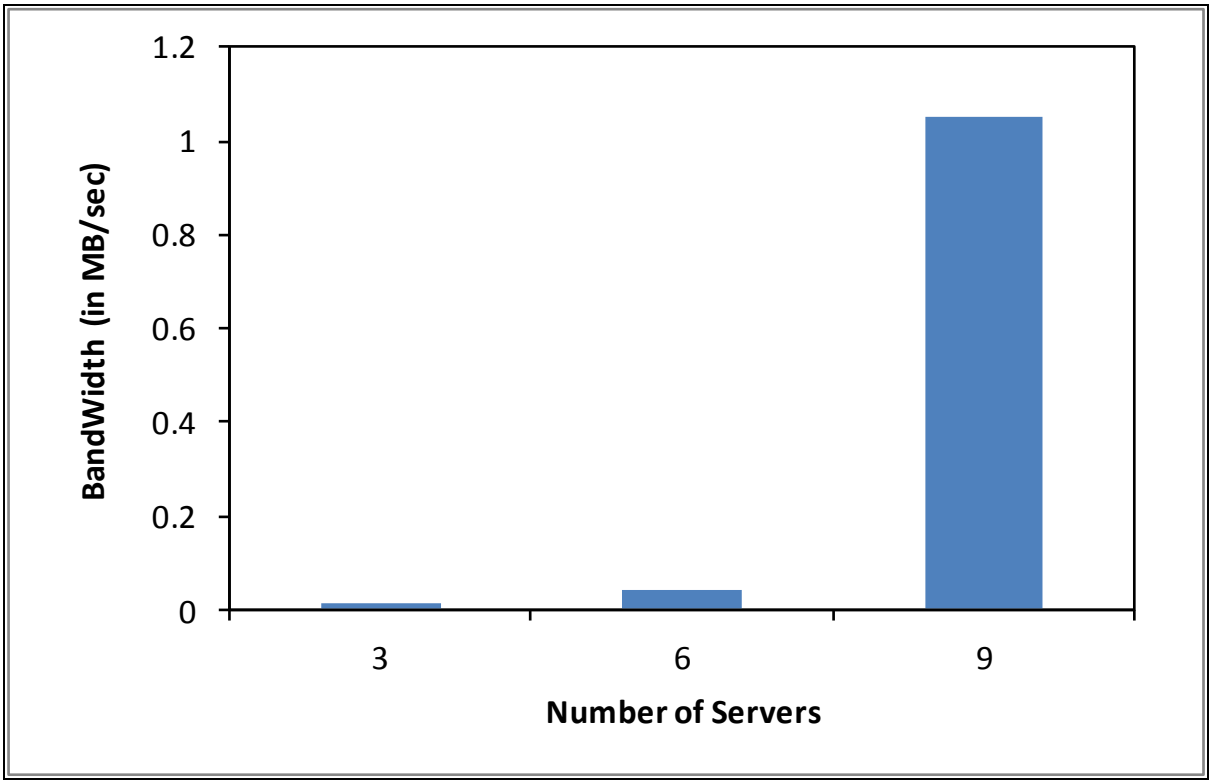

Figure 7.3 - Bandwidth Vs Number of servers

7.3.3 Size of the file

The Android application performance is also measured in terms of time taken to save the file and retrieve the file over the network. The time taken to perform the selected feature also depends on the size of the file. From the following graph, we can say that as the size of the file increases, the time taken to upload more number of chunks to the distributed file servers increases proportionally.

As the size of the file increases, the number of encrypted chunks to be stored on the distributed file servers also increases. Increased number of chunks increases the number of

requests for the Tomcat server to service them. As the number of requests increases, the total time taken to service all the requests received by the server is increasing proportionately.

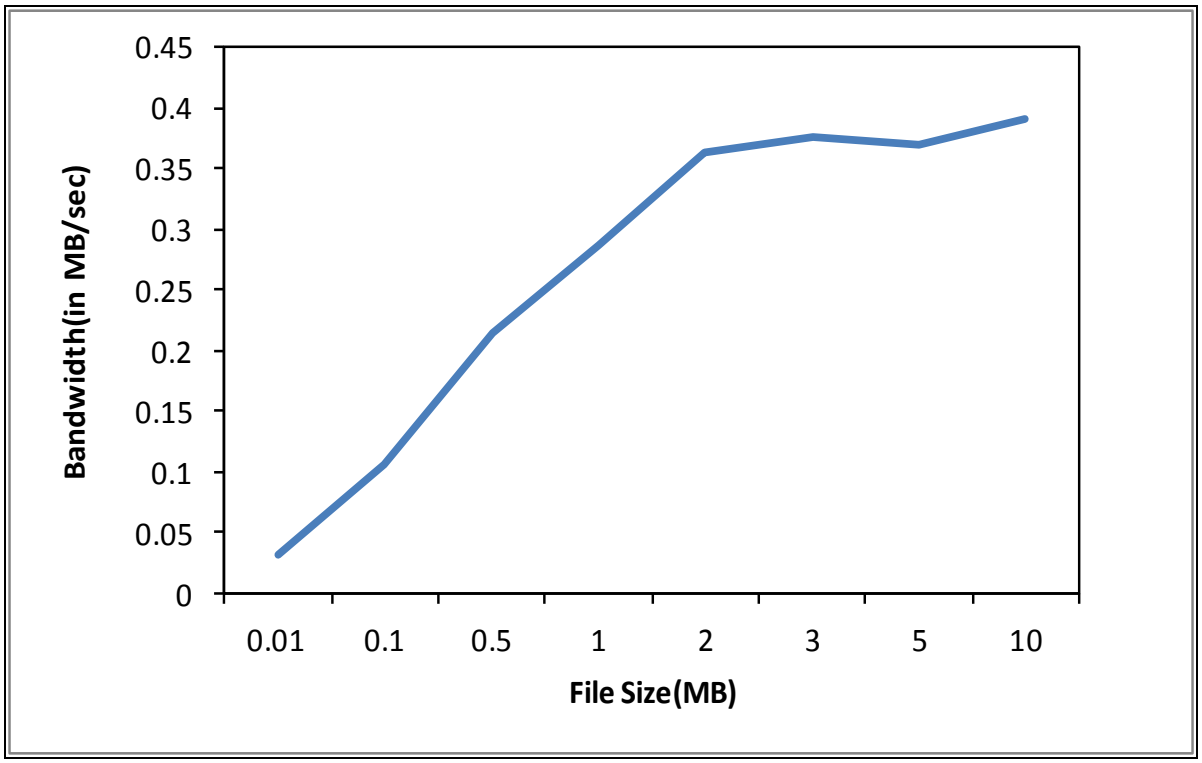

Figure 7.4 - Bandwidth Vs File size

Chapter 8 - Conclusion

The development and working of Android application to transfer the file from the Android device to the distributed file servers as well as the downloading of the original file from the distributed file servers has been elucidated in this project report. Starting from the requirements of the application, each and every phase of the software development life cycle process has been discussed. Given the present-day scenario, people are hardly using laptops to Smartphone for accessing documents. So, there is a need to develop an Android application satisfying the user requirements of storing the file securely on the distributed file servers and retrieving the file when required in the future.

Working with this project helped me to learn more about the Android SDK and the different features available in the versions of Android. The features of this Android application have been changed in the process of the development of an application because of the lack of availability of certain features pertaining to Android platform and the Android devices.

The development of an Android application in combination of distributed file system concept with Android is quite interesting. Incompatibility and lack of availability of certain features during the development of the application helped me to think in a different direction. The Android application has been successfully built in accordance with the current Android device user requirements.

Chapter 9 - Future Work

After implementing the described Android application, the possible enhancements are:

- The duplicated copies present over the inactive server are not deleted when an additional copy is created on another server for all the files present on the inactive server. So, the memory used to store the unused files has to be flushed for future use.
- If a server is inactive and goes down for some time and became active after some time, the server is not being reused. So, inactive servers becoming active servers had to be handled.
- If an Android device identifies another Android device on the network, then instead of using tomcat servers, the Android devices itself can act as server for the direct communication between the two Android devices.

Chapter 10 - References Or Bibliography

[1] Android Development Guide, http://developer.android.com/guide/basics/what-is-android.html

[2] SQLite. http://www.sqlite.org/docs.html

[3] Apache Tomcat , tomcat.apache.org/

[4] http://java.sun.com/developer/technicalArticles/Security/AES/AES_v1.html

[5] Code snippets from stackoverflow - http://stackoverflow.com/

[6] Android Architecture - http://upload.wikimedia.org/wikipedia/commons/6/63/System-architecture.jpg

[7] Unit Testing - http://en.wikipedia.org/wiki/Unit_testing , May 2012.

[8] Integration Testing - http://en.wikipedia.org/wiki/Integration_testing , May 2012.

[9] Server Image - http://news.cnet.com/i/bto/20070108/demediasmartserver_270x347.jpg

[10] Tablet Image - http://www.adobetutorialz.com/content_images/Freebies/Samsung-Galaxy-Tab/samsung-galaxy-tab-preview.jpg

[11] File Image – http://images.all-free-download.com/images/graphicmedium/text_file_icon_55450.jpg

[12] Apache Tomcat Logo - http://webapp.org.ua/wp-content/uploads/2011/11/apache-tomcat-install.jpg