Secure Mobile-Cloud Storage with Location Based File Transfer System

Sribhagat Varma Nampally

Dept. of Computer Science, Arizona State University, Tempe

snampall@asu.edu

Hari Chandana Kanchanapally

Dept. of Computer Science, Arizona State University, Tempe hkanchan@asu.edu

Rachita Satyasi

Dept. of Computer Science, Arizona State University, Tempe

rsatyasi@asu.edu

*Abstract*—Mobile devices in the recent times have occupied a significant part of our lives. Most of the users store their personal information and documents on the mobile device. With the growing size of the information, space on the mobile device has become a major concern. Cloud storage has provided a considerable solution to the space requirement problem on mobile devices. However, this requires fast internet access on the mobile device to store and retrieve the data on the remote cloud. This project proposes a solution to tackle this problem and contains the design, implementation and evaluation of a mobile cloud location-based service (LBS) storage on both mobile device and cloud environment. Since security of the files stored on the cloud can be a problem, it requires the encryption of the user’s data before storing it on the cloud server. In addition to this, RSA digital signatures can be implemented during file transfer between mobile devices over the network [1]. This would verify data consistency and user’s authenticity.

Keywords— Location based file transfer, Security on cloud, Secured transfer between mobile devices, Mobile-Cloud Storage.

# I. INTRODUCTION

Major trends in mobile devices have been noticed with the increase in the usage of cloud storage and faster internet connectivity. As data keeps growing, a major challenge faced by mobile users is storage space, where the users have to prioritize the data they want to store on their mobile devices or find an alternative way to ensure the data availability. Also, not all the data stored on a device is required by the user at all times and hence, having a cloud storage with sufficient space where user’s data can be stored securely and fetched on demand can be a suitable solution to the space constraint on the mobile devices [2], [3], [4]. This also reduces the cost of investment on large storage spaces.

Retrieving a file from remote cloud requires more bandwidth and takes more time based on the accessibility. Location based services on mobile devices can address this problem by sending the requested file from an authorized mobile device nearby [2]. In this approach, the mobile device that requires a local copy of the file uses the location based services to determine a nearby mobile device which is authorized to transfer the file to it. Once this nearby mobile device is determined, it can directly interact with it to complete the file transfer.

In addition to these problems, there are high chances of attackers impersonating themselves in the network and also trying to manipulate user’s sensitive data during transit. The use of Asymmetric keys to encrypt and decrypt messages at sender and receiver’s end respectively would help keep a check on the security risks during transfer [1].

In order to tackle all these problems, we propose a secure mobile-cloud storage with location based file transfer system where the data is encrypted before it is stored in the cloud. We propose an innovative way to retrieve the files when requested. This is based on the location-based services available on the mobile devices where we check for the availability of the desired file on the mobile phones within the proximity of the requested mobile’s location and allow it to take a copy of the file from the nearby mobile device only if it is authorized to access it [2]. In this project, we aim to build a system that has a cloud server to store user’s data and an android application with rich GUI that can upload a file to the secure cloud server [5] and request a file from the cloud or a nearby mobile device based on the concept mentioned above. We also impose security to the data by using public key cryptographic algorithms and RSA algorithm with digital signatures to authenticate and authorize the users respectively.

# II. System Models

## System Model

Consider two mobile devices (Mobile 1 and Mobile 2) available at different locations.  Each mobile device has a set of files with them. Some of these files are backed up on the cloud based on the user’s preferences. In order to store the file in cloud, the mobile device (say Mobile 2) has to register itself onto the cloud. The cloud then authenticates the user of Mobile 2 and stores the file.

Now, if the Mobile 1 requires a local copy of a file, it uses the location-based services to check the availability of an authorized mobile device, which contains the required file on its local storage. Once determined, the Mobile 1 can directly request for the transfer of the file thereby reducing the burden on the cloud. The Mobile 2 then encrypts the file using its private key and then with the public key of Mobile 1 and initiates the file transfer. If there is no mobile in the proximity of Mobile 1, request goes to the cloud where it checks for the availability of the file in its storage. If present, the cloud server encrypts the file and sends it to the requested mobile device.

Decryption is done at the receiver end depending upon whether the file is been sent from the cloud or the mobile device.

## Software

The environment set up to develop the application is:

- Operating system: Android

- Cloud environment:

- Language of implementation- Java

- Database:  SQL server

## Security Model

We use public key cryptography (RSA algorithm) to encrypt the files before the file is stored in the cloud [1]. This model prevents the data access if the message is hacked by the intruder, as he is not aware of the keys used to decrypt the message. We plan to use RSA digital signatures algorithm to encrypt the files while transferring it between the mobile devices. This would ensure the authenticity of the user as the sender signs the document with its private key and then encrypts the message with the public key of the receiver before sending the file to the destination. The receiver decrypts the message with its private key; this would ensure that the message is intended to it. Then it decrypts the message with the public key of the sender to verify that the message has originated from the right source.

This security model tackles ‘man in the middle’ attacks and ensures that the intruder doesn’t have access to the file.

# Project Description

## Project Overview

The users of the system are ‘Mobile 1’, ‘Mobile 2’ and the ‘Cloud’. The project can be divided into two primary tasks: ‘storing the file onto the cloud server’ and ‘request and transfer of the file from mobile or cloud.’

Mid-term goal of the project is as following:

-  Upload file onto the cloud server.

-  Request a file for download.

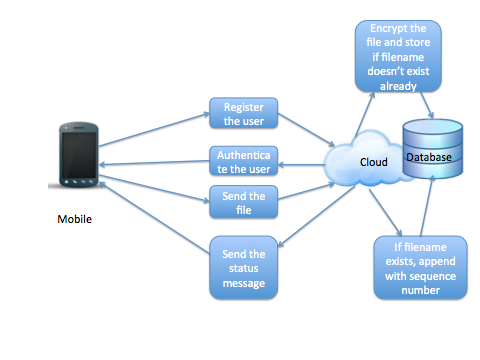
-  Set up a cloud server that can handle the request and transfer the file either from the cloud storage or facilitate a mobile device nearby to transfer it.

Final goal of the project is the following:

-  Build a secure file transfer system either from the mobile or cloud using RSA and digital signature encryption techniques.

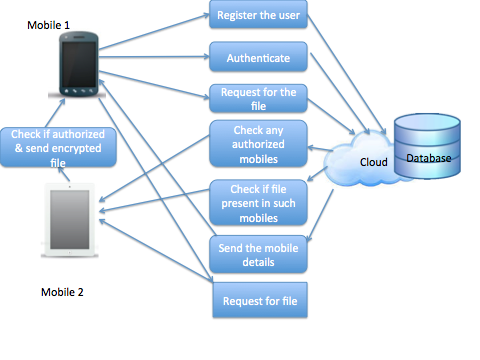
-  A system that allows file transfer only between authenticated users (friends of users)

## Task 1 : Storing the file onto the cloud server:



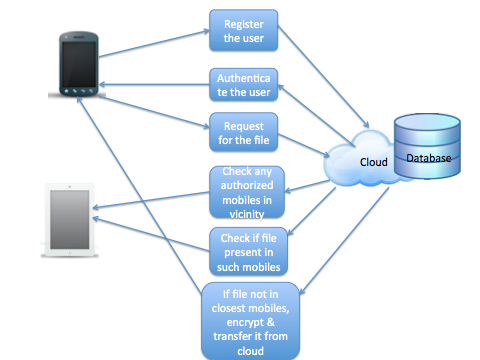
A mobile device authenticates itself and then uploads a file onto the cloud server. The cloud server checks the existence of a file with the same name in its storage and then stores the file. If a file exists with the same name, the file is handled by appending the sequence number to it. We encrypt the file before sending it to the cloud to ensure data security.

*Task 2: Request and transfer of the file from mobile device*



The mobile device that needs a local copy of a file searches for any authorized mobile devices around it. It also checks if these authorized devices have a local copy of the requested file. If present, the device would directly initiate a request to one of the authorized mobile devices for the transfer of the file. This authorized mobile device would then send the encrypted file to the requesting mobile device.

## Task 3: Request and transfer of file from cloud



If no mobile device is available in vicinity, or if the mobile devices in vicinity do not contain a local copy of the required file, the cloud server encrypts (with public key of receiver) and transfers the file to the mobile device requesting it.

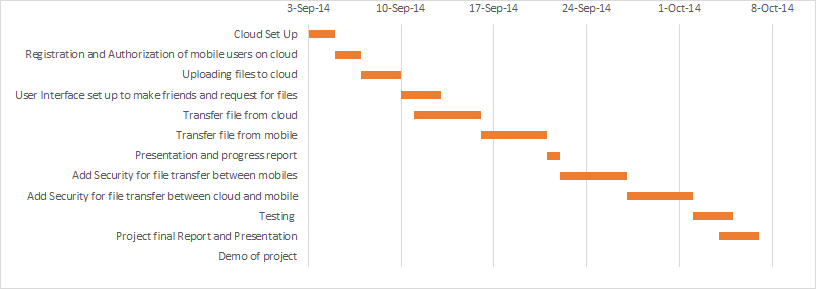
## Project Task Allocation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Allocation** | **Workload** | **Team Member** |
| Cloud set up | 2 | 5.4% | Rachita |
| Registration and Authorization of mobile users on cloud | 2 | 5.4% | Chandana |
| Uploading files to cloud | 3 | 8.1% | Rachita |
| Managing the MySQL database |  |  | Chandana |
| User Interface set up to make friends and request for files | 3 | 8.1% | Chandana |
| Identify the location of the nearby mobile devices |  |  | Sribhagat |
| Transfer of files between mobile devices |  |  | Sribhagat |
| Downloading files from cloud | 5 | 13.5% | Rachita |
| Transfer file from mobile | 5 | 13.5% | Chandana |
| Presentation and progress report | 1 | 2.7% | Rachita, Sribhagat, Chandana |
| Add Security for file transfer between mobiles | 5 | 13.5% | Sribhagat |
| Add Security for file transfer between cloud and mobile | 5 | 13.5% | - |
| Testing | 3 | 8.1% | Chandana, Rachita & Srbhagat |
| Project final Report and Presentation | 3 | 8.1% | Rachita, Sribhagat, Chandana |

## Deliverables

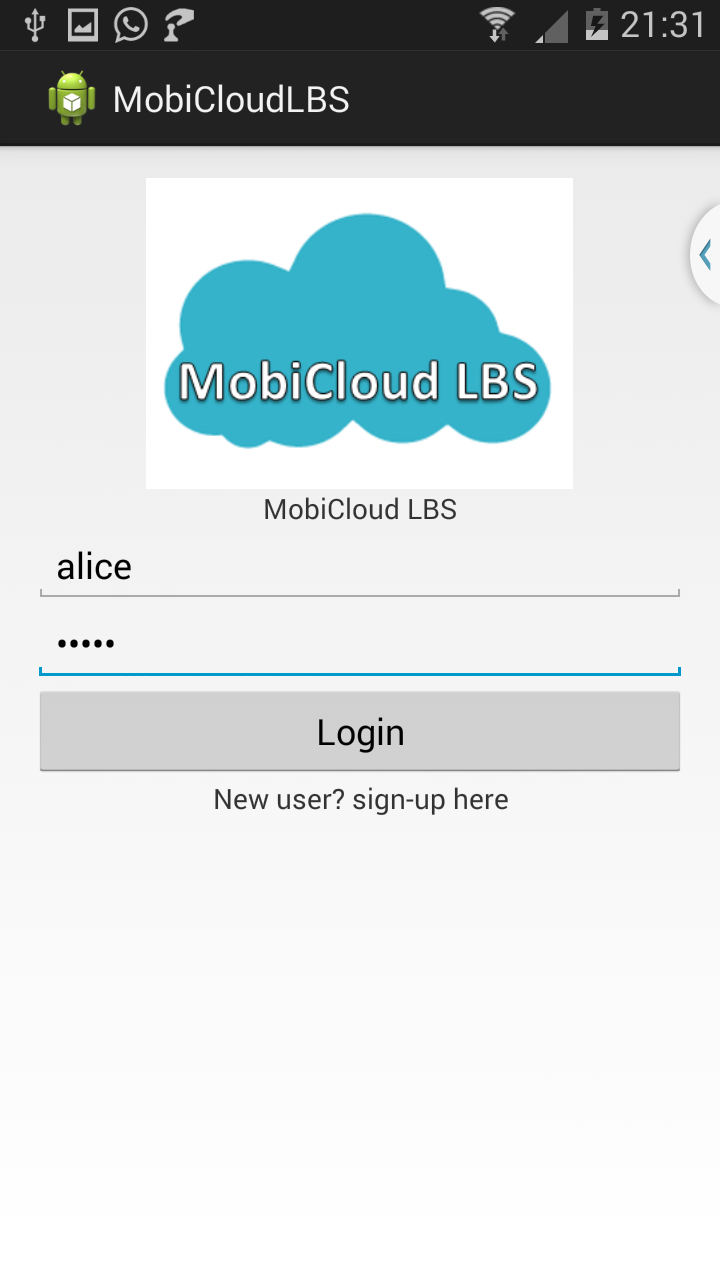
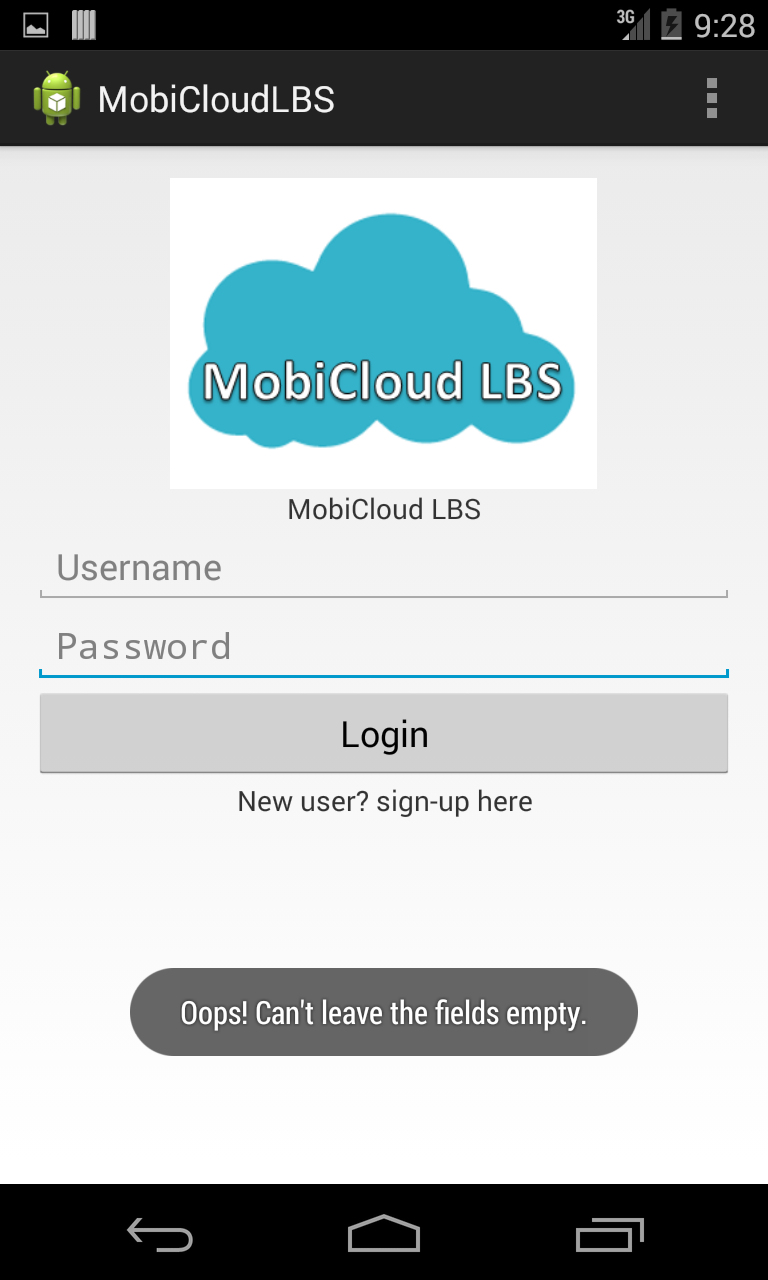
We aim to setup up a cloud environment, which has sufficient storage capacity to save user’s files. We also develop an android application with rich GUI that stores the files on the cloud upon authentication. It also implements location based services and downloads the file from a nearby mobile device or the remote cloud based on the distance and availability. Considering the security issues faced in data storage in the cloud environment, we intend to encrypt the files before storing them on the cloud and also while transferring the files between the mobile devices. This would ensure that only authorized users are allowed to access authentic data.

## Project Timeline



## **IMPLEMENTATION**

***Login:*** The application allows the user to login to the system by checking against the database for registered users. A newuser can navigate to the sign up activity for registering himself onto the system. We have implemented the web-service using Apache, server side validation using PHP and MySQL database to store the information. A new logo for the home screen is added to the home screen

Figure(1) Login Screen Figure(2) Login Screen Validation

***Registering the mobile device on the cloud:*** The user registers into the cloud server through the register screen in the application. This is done by entering the full name, username, e-mail id, the list of friends and password. The values entered will be inserted into MySQL database of XAMPP-a distribution of Apache, MySQL, Perl and PHP. PHP scripts have been written to perform the server side validation and process the register request and store the information in the database.

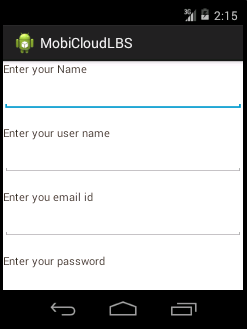
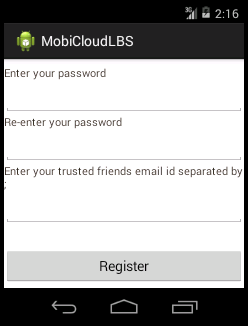
 

Figure (3) Registration page 1 Figure (4) Registration page 2

***Client side validation for the registration screen:*** Application-side validations on username and password are done at the mobile side and the user must adhere to them in order to register into the applications database present on the cloud.

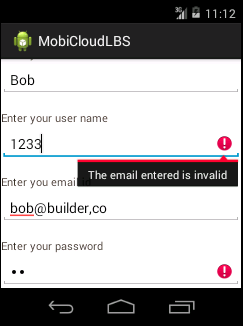
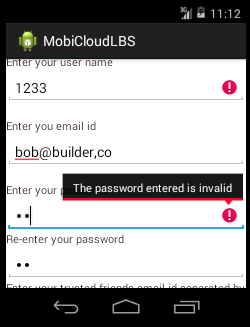
 

Figure (5) Registration page validation Figure (6) Registration page validation

***Uploading files:***After the user logs into the application, the list of files available on the mobile device are displayed. The user can upload the files into the cloud server by clicking the filename. We assign a path in the cloud for this file and store that file path in the MySQL database which will be useful while requesting for downloading the data. Upload button is added to the top of the home screen and the message passing between the intents is implemented to retrieve some of the essential and useful information like the user name, the name of the file that is being uploaded etc.

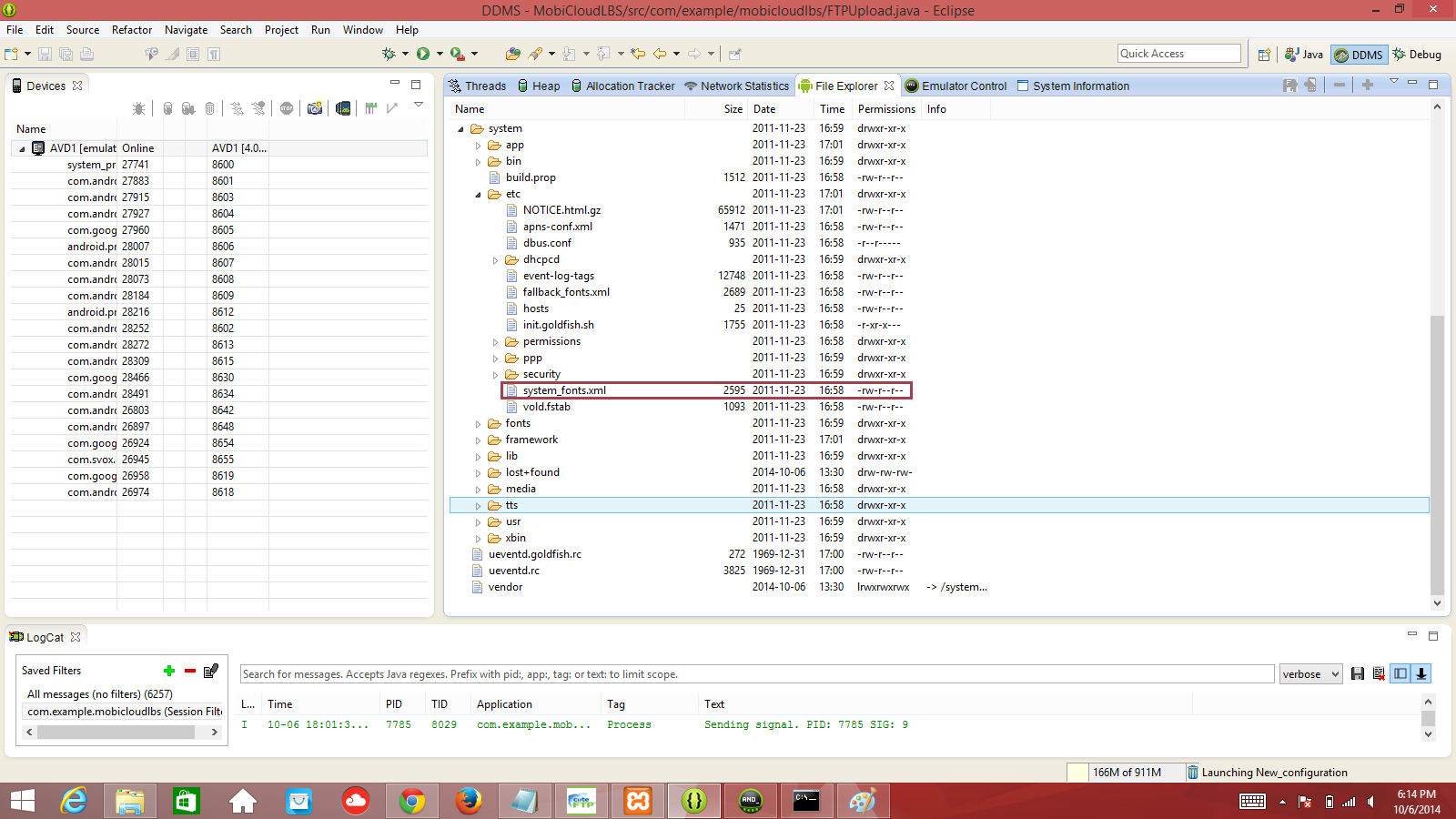


Figure (7) File present in the emulator’s internal storage

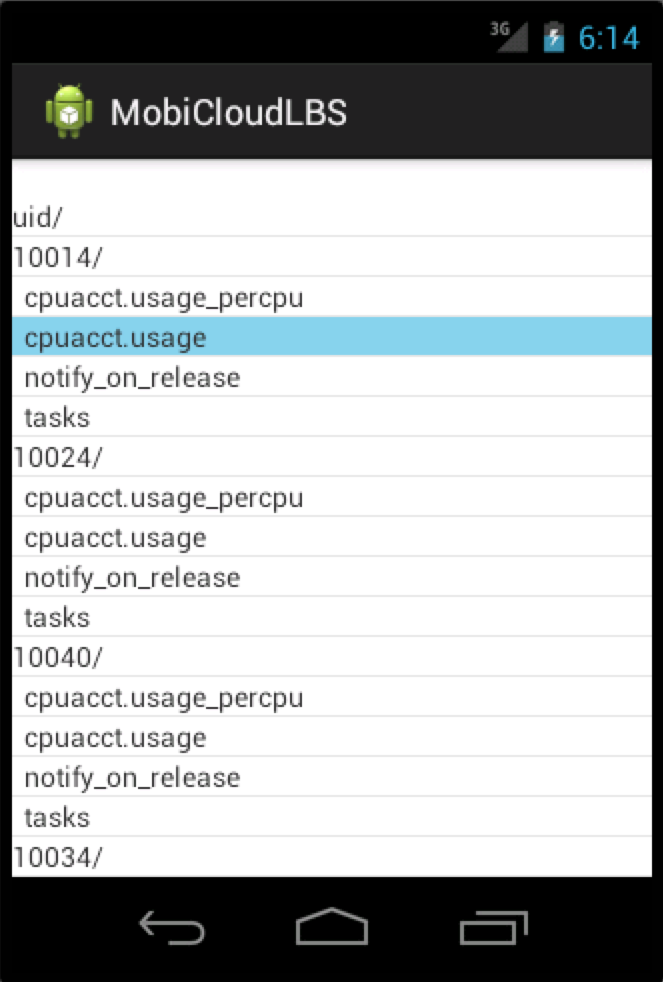
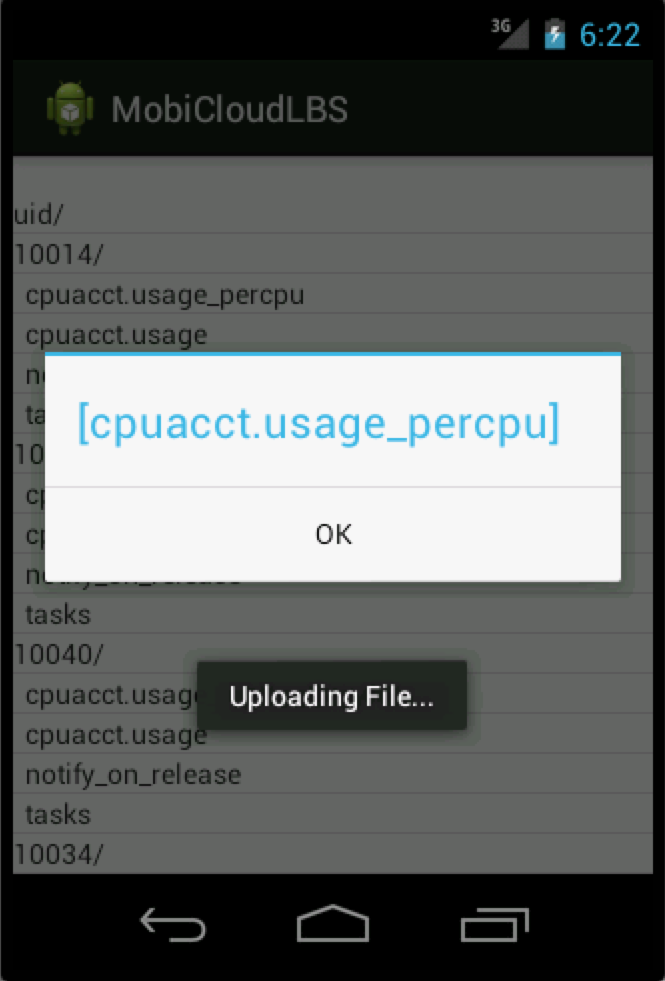
 

Figure (8) Browse the internal storage on phone Figure (9) uploading a specific file name to the database

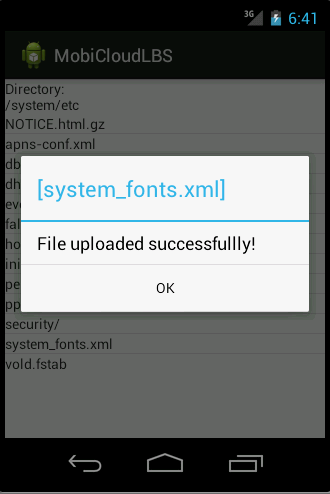


Figure (10) Success message after upload is complete

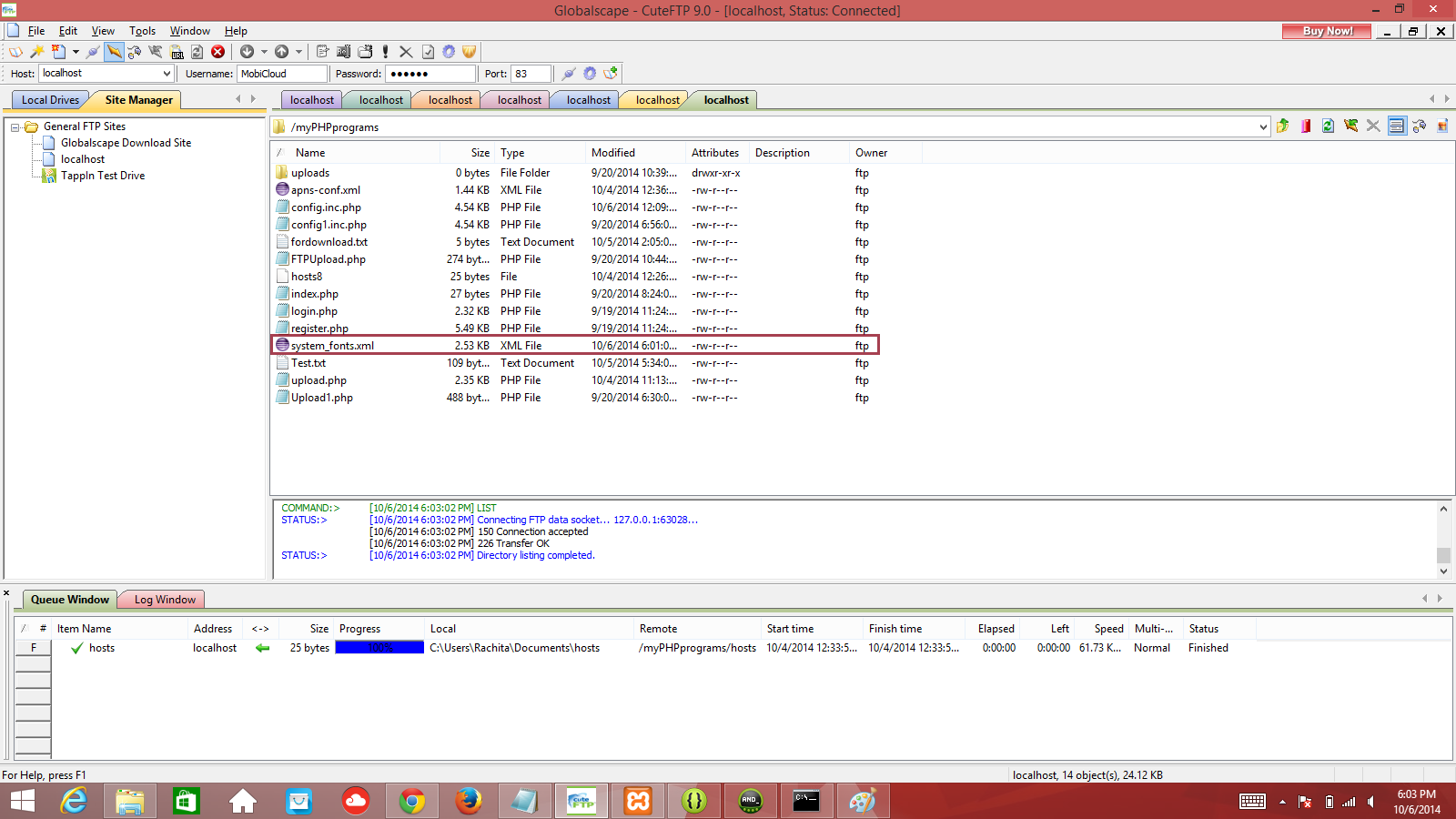


Figure (11) File’s presence on the FTP server

***Download/Transfer Files:***The requesting mobile device checks for the file it requires from the list and clicks on the button to either request for file transfer from a nearby device or download from the cloud.

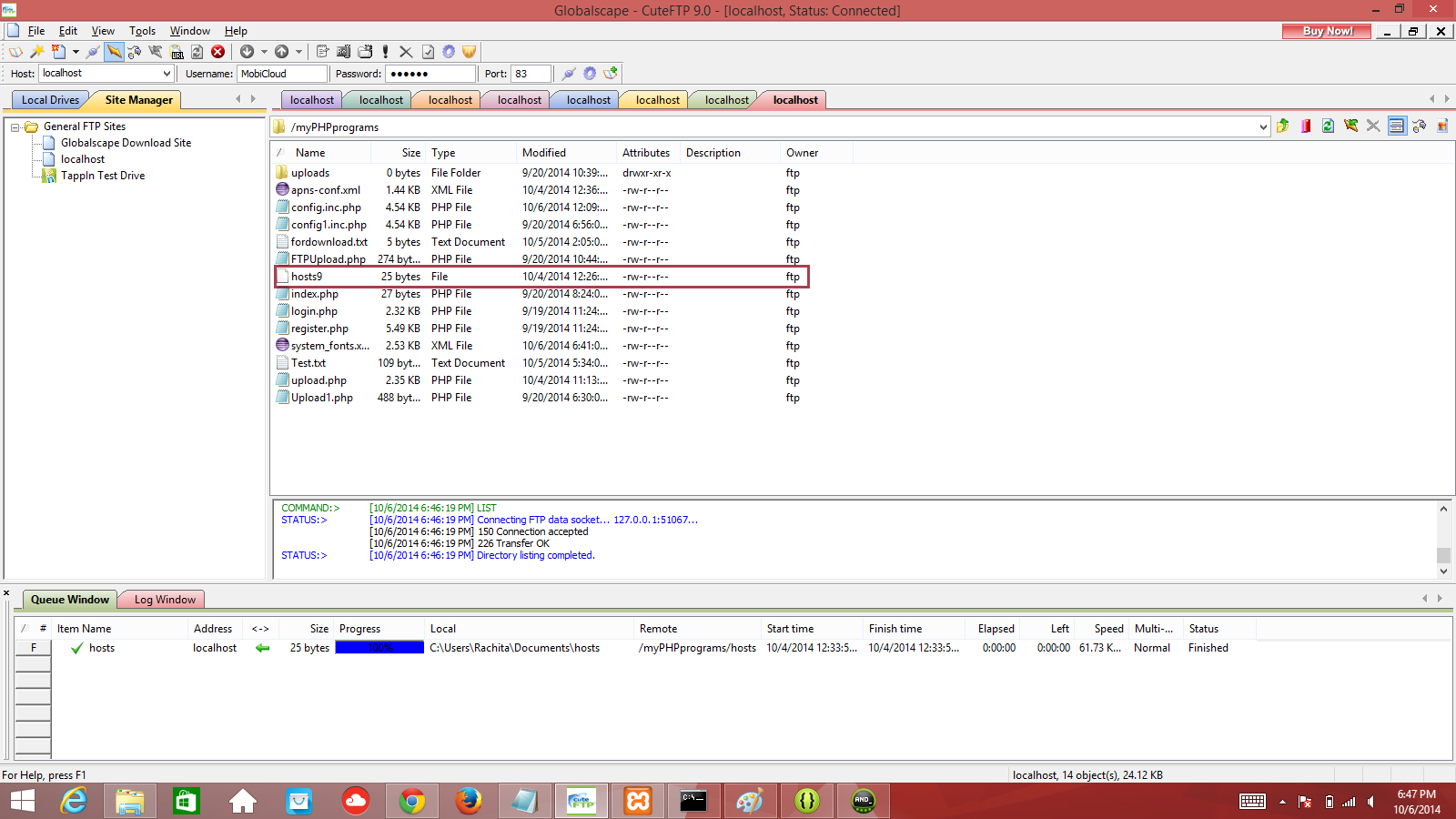


Figure (12) File’s presence on the FTP server

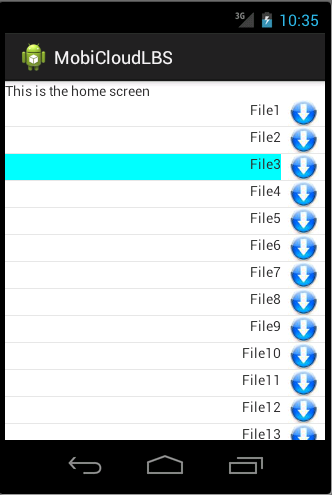
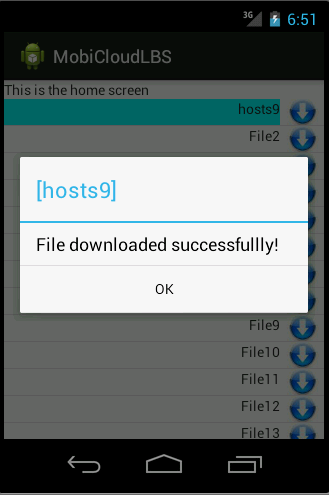
 

Figure (13) Selecting a file from the file’s the Figure (14) File download complete

device has uploaded on the cloud server

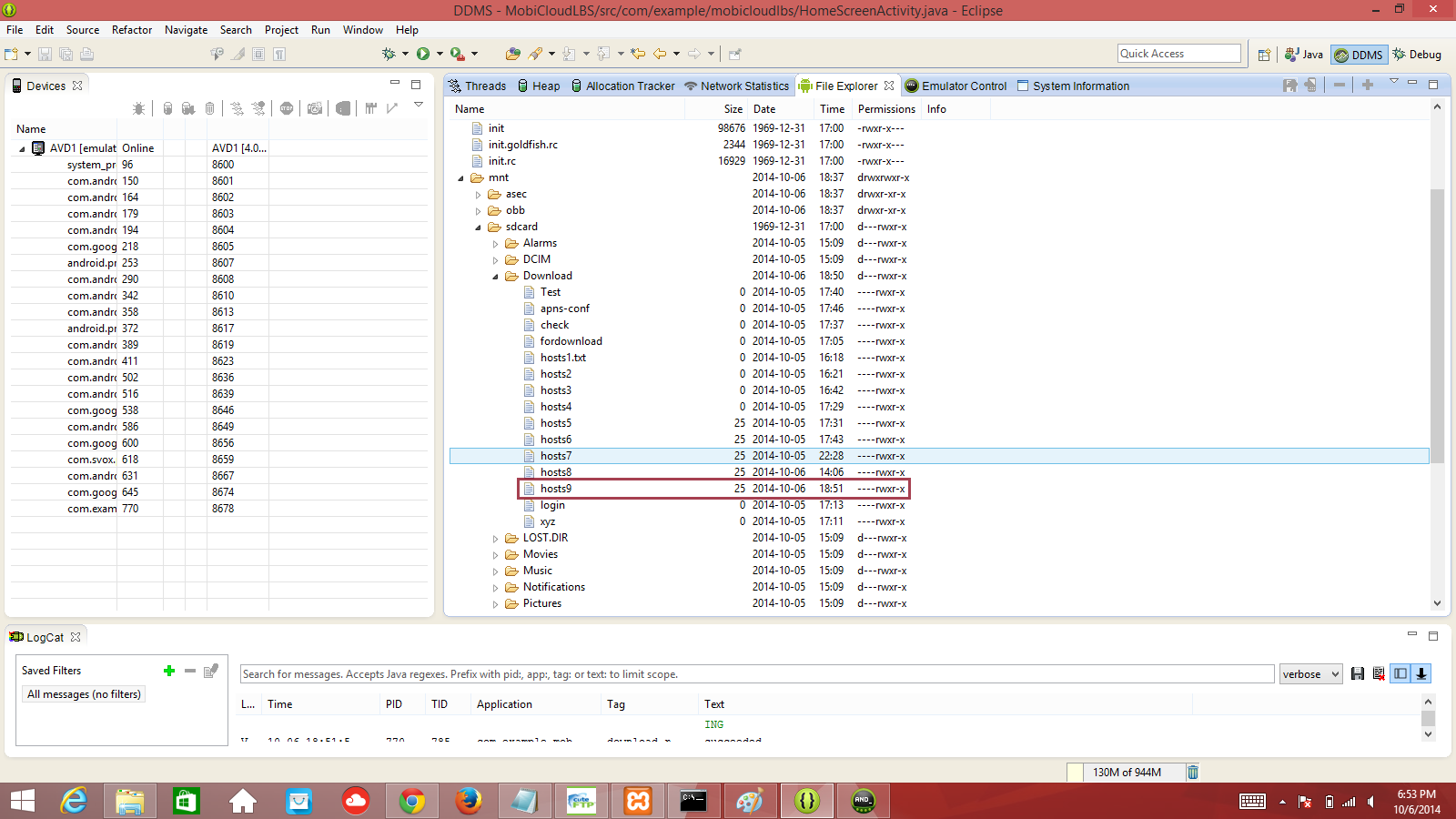


Figure (15) File’s presence on the mobile phone’s internal storage after the download

Once the user is logged into the application, the files present in his cloud are displayed and the availability of the files in his local store are showed. When he needs a file in the local store, the file is either downloaded from the nearby mobile device or from the cloud. Downloading the file from the FTP server is implemented using FileZilla which is integrated into XAMPP.

***Location Discovery and File Transfer between mobile devices:***The core requirement to transfer the file between mobile devices or finding the location of the nearby mobile devices is to establish a connection between them. Socket programming is one of the most widely used way to perform wireless communication between mobile devices. A device can run as a server and listen for the connections while another device can run as a client and connect to the other mobile device that is acting as the server by providing its IP address. Then the mobile devices can communicate with each other.

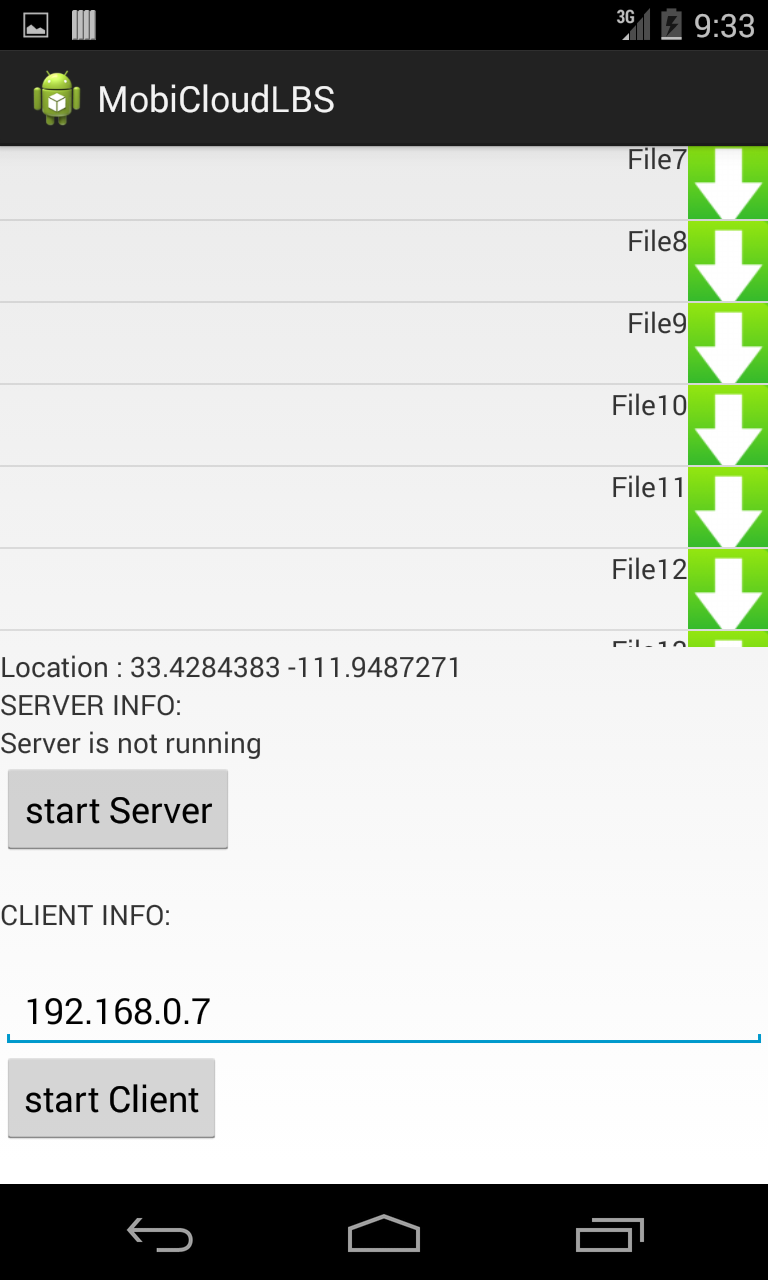
 

Figure (16) Selecting a file to get it from Figure (17) Connecting to the server from client

the nearby location (Sending a file to the requested mobile)

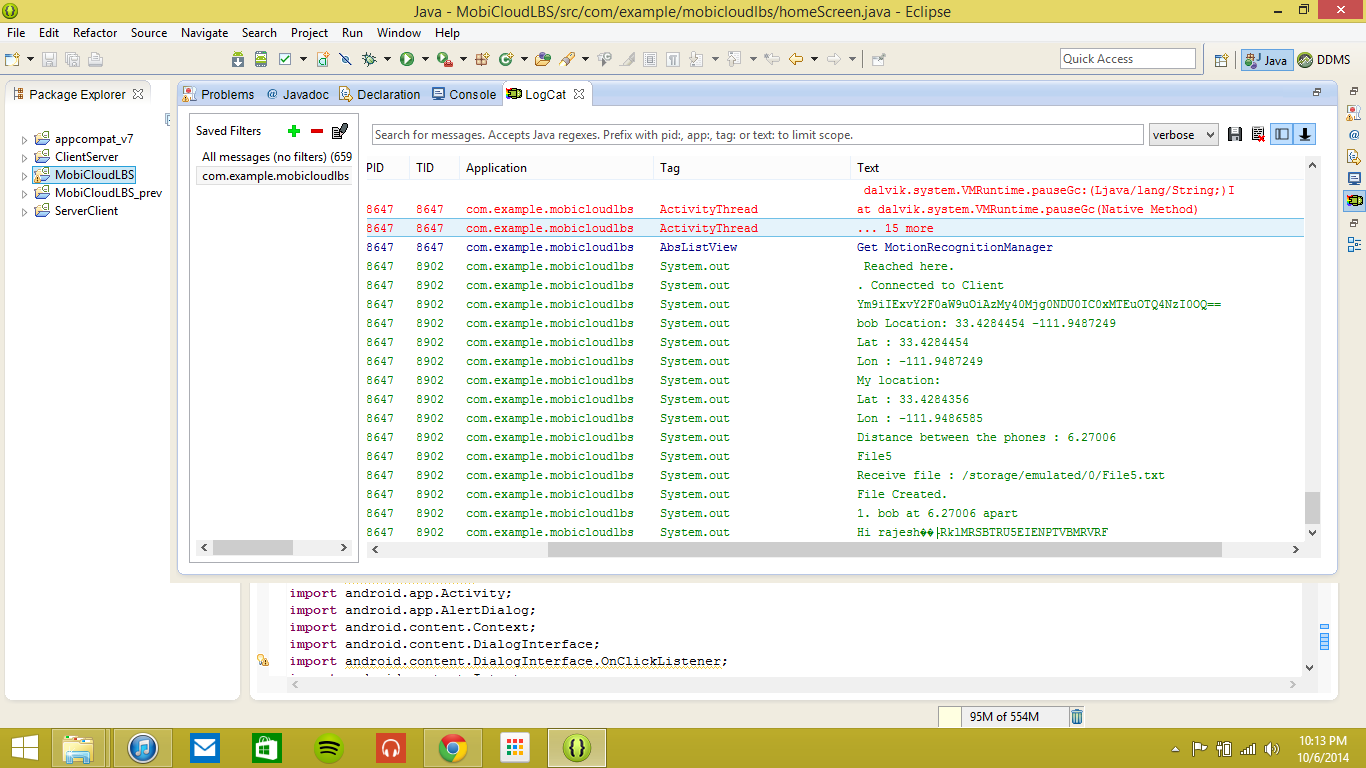


Figure (18) Messages passed between the mobiles. The location of the nearby mobile device, the name of the person and the distance between the mobile devices, also, the content of the file that has been transferred.

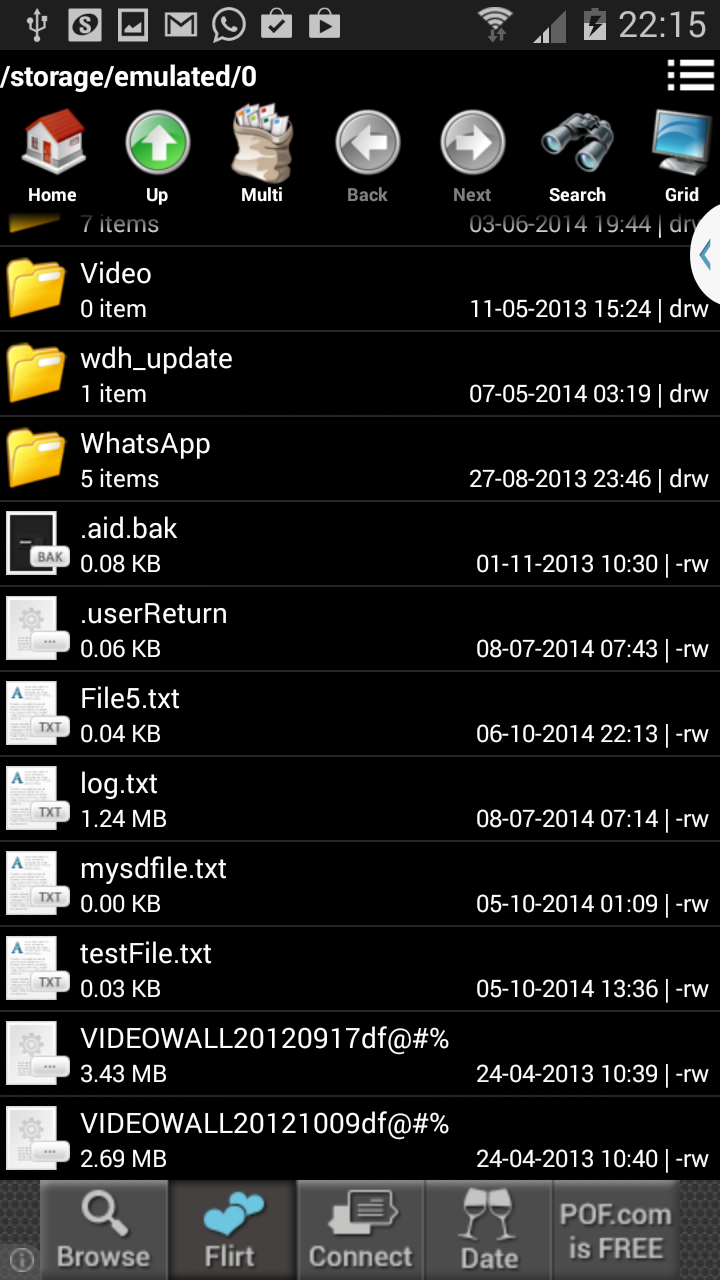


Figure (19) File presence in the mobile 1 (this device sends Figure (20) File’s presence in the mobile 2(after

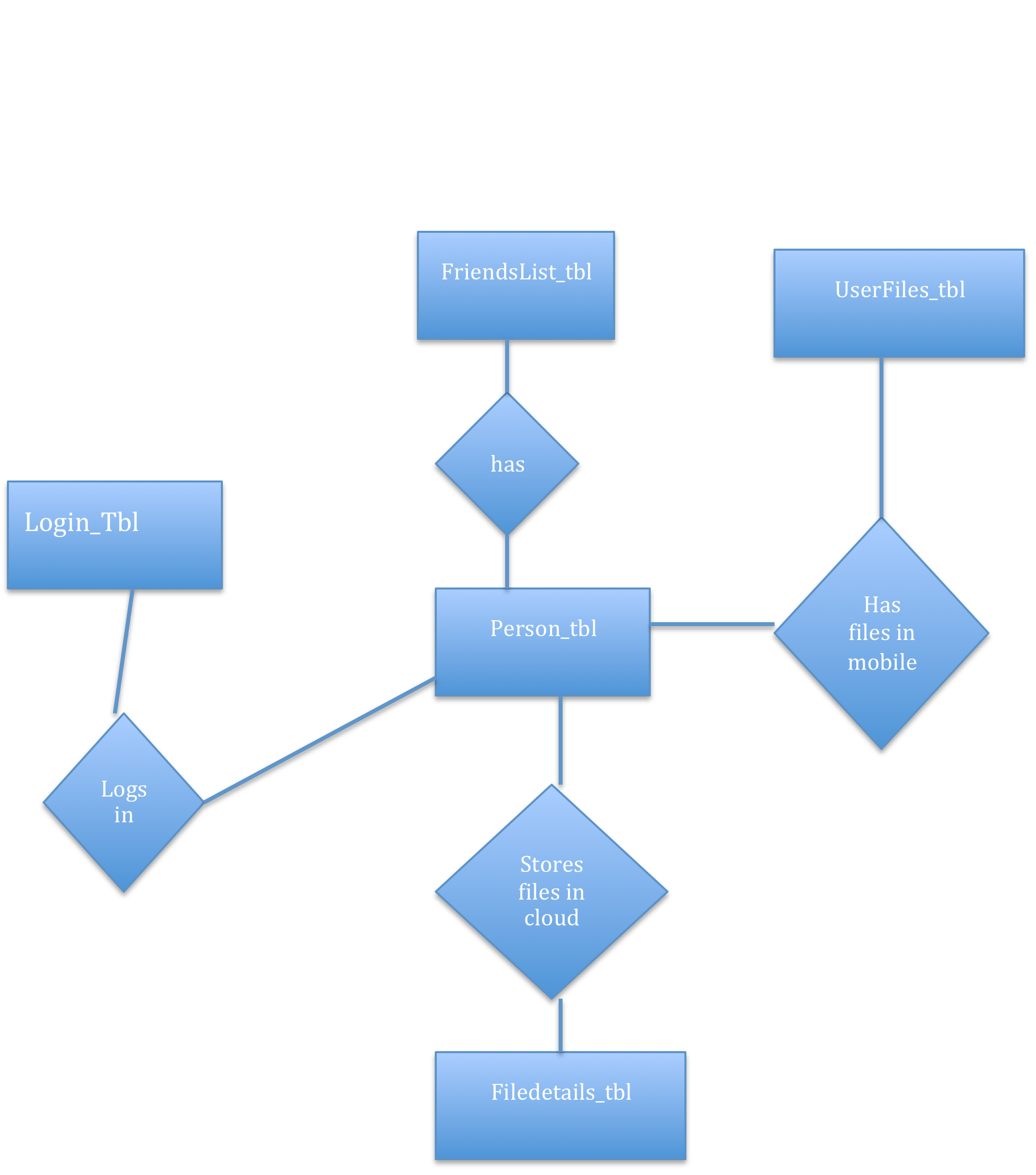
the file to the requested device) downloading from the device it requests)

The functionality of the file download from the server is implemented in this phase using sockets. The mobile device that needs a file on its local storage clicks on a file that it needs. Then it opens it ServerSocket and listens for the accept message from the other mobile devices that act as clients. All the mobile devices respond to by sending their name, availability of file and their respective location. The mobile device then checks if the location is nearby and if the device contains the file. Depending on the message it receives, it either takes the file from the nearby mobile phone or downloads the file from the FTP server.

***Security:***

RSA algorithm is one of the widely used protocols for ensuring authentication and non-repudiation. Initially we have begun implementing RSA algorithm for the message communication between mobile devices. The algorithm has been implemented, but the results found that the encryption didn’t lead to the same message as the original message. Due to the constraint of time, simple android.util.Base64 library to encrypt and decrypt the messages. This is a very simple yet very fast way of encrypting and decrypting data on Android platform.

***ER Diagram:***



***Challenges:***  
*Communicating with the Cloud Environment:*

Communication with the cloud environment has to be done with its public IP that has been provided to us. But, invoking the cloud from a local machine or an emulator has been a challenge. For this purpose, we have used the XAMPP which supports MySQL database, PHP and simulated the functionality of exposing the services on the local machine.  
*Identifying the IP of mobile device to transfer data:*

Evaluating a way to transfer data between the mobile devices without the knowledge of IP address of the receiver. We would like to make a socket communication between the mobile devices initially and then transfer the files between them using IP’s. This can be extended to transfer the data between the mobile devices without knowing their IP, which can be an ideal case in the real world. We are looking for effective solutions to do this. Modifications to the existing idea would be in the future.  
*Discovery of nearest mobile devices:*

Calculating the distance between the mobile devices can be done through identifying the nearest mobile device from source. This leads to a lot of computation if the number of mobile devices in the network is very high.

*Communicating with the mobile devices using Sockets:*

Proper use of output streams and synchronization between the client and server messages are needed for effective communication between the mobile devices. Having a standard format for messages is therefore needed to pass some important information like name of the user, location of the client, device and availability of file in its local store. The next phase is the transfer of files between the mobile devices. This involves identifying the folder structures, the files and the folder structure at the destination. For testing the application, we have used .txt file for sending and receiving between the mobile devices.  
*Implementing Security:*

Implementing security by using RSA algorithm with digital signatures requires tremendous effort in understanding the algorithm and the data structures that are needed to implement this algorithm. Even though this was implemented, the resulting code was not efficient in processing the message and generate proper encryption and decryption keys. Hence, we had to look for alternative ways to implement encryption and decryption to ensure the security of the data transferred from the application.

Unable to download files of different formats accepting only a particular format of files to be downloaded into the SD card.

Making the SD card on the emulator writable was a challenge.

# Risk Management of the project

Ensuring high level of security by providing authentication and authorization to the users is a complex task and requires tremendous effort. As there are many implementations of these techniques and considering the time available to implement our project design, we plan to concentrate on the core functionalities of this project design i.e., storing the file on the cloud storage, retrieval of the file using location based services and encryption of the data during these processes.

# Conclusion

Creating a rich GUI android application, which uses cloud services to store the user’s personal information, alleviates the space problems faced by traditional mobile devices. The addition of location-based services to download the data from the surrounding authorized mobile devices reduces the network bandwidth and download time of the file. This would also reduce the burden on the cloud server as the mobile device requesting the file checks for an authorized mobile device without interacting with the cloud. This project can be extended to provide broadcasting feature where a professor can share files to a set of students who are registered to a specific course under him.

We have specified a static value to calculate the proximity of the mobile devices. A significant amount of research can be done in the future to calculate the dynamic threshold for proximity. This requires analysis of the data captured viz. file transfer rate and the time taken for file transfer between the mobile devices under different networks and comparing them with the same attributes captured between the mobile device and the cloud. A plot of the collected results would help us arrive at a probable threshold of proximity.

##### Acknowledgment

We thank Professor Dijiang Huang for the knowledge he has imparted to us and for giving us the opportunity to work on the project. We also thank Mr. Tianyi Xing and Bing Li for their constant support and guidance. We also extend our sincere thanks to ASU for providing us with the infrastructure and facilities to develop the project.

##### References

1. Preeti Garg, Dr. Vineet Shanna, “An Efficient and Secure Data Storage in Mobile Cloud Computing through RSA and Hash Function”, 20J4 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)
2. Patrick Stuedi, Iqbal Mohomed, Doug Terry, “WhereStore: Location-based Data Storage
3. for Mobile Devices Interacting with the Cloud”, MCS’10, June 15, 2010, San Francisco, USA
4. Nazir Ahmad Dar and Afaq Alam Khan, “[A System to Track Android Devices- An Implementation of LBS, Location Manager, Services and Web-Services in Android](http://www.academia.edu/5745851/A_System_to_Track_Android_Devices-_An_Implementation_of_LBS_Location_Manager_Services_and_Web-Services_in_Android)”,SST Journal of Mathematics & Computing System, Vol. 4 No. 1, (January - June 2013), p.p. 49-54 ISSN No. 0976-9048 © Intellectuals Society for Socio-Techno Welfare
5. Manav Singhal, Anupam Shukla,”Implementation of Location based Services in Android using GPS and Web Services”,K237 IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012
6. <http://wowjava.wordpress.com/2011/01/16/login-application-for-android/>
7. <http://msdn.microsoft.com/en-us/evalcenter/dn434042.aspx>
8. <http://www.javacodegeeks.com/2010/09/android-location-based-services.html>
9. <http://catchmayuri.blogspot.com/2011/05/get-current-location-gps-position-in.html>
10. <http://www.codota.com/android/scenarios/52fcbdd6da0a6fdfa462f0d3/finding-current-location?tag=dragonfly>
11. <http://developer.android.com/training/connect-devices-wirelessly/nsd.html>
12. <https://android.googlesource.com/platform/development/+/master/samples/training/NsdChat/src/com/example/android/nsdchat/NsdHelper.java>
13. <http://justanapplication.wordpress.com/2013/10/29/service-discovery-in-android-and-ios-part-one-the-android-net-nsd-nsdmanager-class-considered-not-very-useful-at-all/>
14. <http://www.w3.org/TR/discovery-api/>
15. <http://www.androidhive.info/2011/10/android-login-and-registration-screen-design/>
16. <http://android-er.blogspot.com/2010/01/implement-simple-file-explorer-in.html>
17. http://stackoverflow.com/questions/1720346/how-to-get-the-android-emulators-ip-address
18. <https://github.com/ParsePlatform/PushTutorial/tree/master/iOS>
19. <http://www.mybringback.com/series/android-intermediate/page/4/>
20. <http://www.gigarank.net/free-hosting.php>
21. <http://wiki-android.blogspot.com/2012/12/creating-android-ftp-client-ftp.html>
22. <http://stackoverflow.com/questions/8678364/android-ftp-client-application-how-to-login-into-the-ftp-server-what-ftp-serve>
23. <http://androiddevblog.blogspot.com/2010/08/android-ftp-client.html>
24. <http://androidexample.com/Upload_File_To_Server_-_Android_Example/index.php?view=article_discription&aid=83&aaid=106>
25. http://developer.android.com/reference/android/util/Base64.html
26. http://developer.android.com/reference/java/net/Socket.html
27. http://developer.android.com/reference/java/net/ServerSocket.html
28. http://developer.android.com/reference/java/io/File.html
29. http://developer.android.com/reference/java/io/InputStream.html
30. http://developer.android.com/reference/java/io/OutputStream.html
31. http://developer.android.com/reference/android/location/LocationManager.html
32. http://developer.android.com/reference/android/location/LocationManager.html
33. <http://developer.android.com/reference/android/location/LocationListener.html>
34. [http://androidtrainningcenter.blogspot.com/2014/02/android-ftp-client-tutorial-with.html](http://www.google.com/url?q=http%3A%2F%2Fandroidtrainningcenter.blogspot.com%2F2014%2F02%2Fandroid-ftp-client-tutorial-with.html&sa=D&sntz=1&usg=AFQjCNGhaAWUHTtTM8-pFYP6kDihzvkr2w" \t "_blank)
35. [http://www.egeek.me/2012/11/22/downloading-a-file-via-ftp-in-android/](http://www.google.com/url?q=http%3A%2F%2Fwww.egeek.me%2F2012%2F11%2F22%2Fdownloading-a-file-via-ftp-in-android%2F&sa=D&sntz=1&usg=AFQjCNHH8g2Z_rjWNWBtnLEqjWb1bzndaQ" \t "_blank)
36. [https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/util/Base64InputStream.java](https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/util/Base64InputStream.java" \t "_blank)
37. [http://stackoverflow.com/questions/2083709/android-emulator-sdcard-push-error-read-only-file-system/12797483#12797483](http://www.google.com/url?q=http%3A%2F%2Fstackoverflow.com%2Fquestions%2F2083709%2Fandroid-emulator-sdcard-push-error-read-only-file-system%2F12797483%2312797483&sa=D&sntz=1&usg=AFQjCNGAdHzNjrT1wostzyG5COsu5soigQ" \t "_blank)
38. [https://www.codota.com/android/scenarios/52fcbca6da0a8843576c6e10/android.util.Base64OutputStream?tag=dragonfly](https://www.google.com/url?q=https%3A%2F%2Fwww.codota.com%2Fandroid%2Fscenarios%2F52fcbca6da0a8843576c6e10%2Fandroid.util.Base64OutputStream%3Ftag%3Ddragonfly&sa=D&sntz=1&usg=AFQjCNFPN4yFRbjVa9Pn4qPABzB0fmRPdA" \t "_blank)
39. [https://www.youtube.com/watch?v=x8xEZq43wgc](https://www.youtube.com/watch?v=x8xEZq43wgc" \t "_blank)