Overview of Secure File Share Over Mobile Network

Mohd Kamil Hussain, Siddhartha Sankar Biswas*, Safdar Tanweer, Zunaid Aalam
{* Corresponding author : ssbiswas1984@gmail.com}
Jamia Hamdard, New Delhi - 110062

Abstract. Secure file sharing is application that enable sharing of files between mobile devices with least involvement of external http server. Generally QT is a framework used for development of embedded applications such as Elevator control panel, Washing Machine program selector and Router management system etc. With release of QT5 the domain of application development has broaden to personal computers and mobile devices.I have used this framework to develop and generate apk for this application.The core part of this study has been dedicated to development of communication model and defining a protocol and flow of file sharing process, I have also developed some custom components to fill the gap of QT framework in realm of android application. I have developed light encryption method and authentication mechanism to keep the file sharing more secure.

Keywords: Application workflow, Server component, Client Component, Listener Cycle

1 Introduction

The practice of sharing digital media such as documents, multimedia (audio/video), graphics, computer programs, images and e-books etc is the primary requirement of any information system. The main reasons for motivation behind the selection of this study is that traditional file sharing applications rely on a server to share data between devices on different networks. The server is a middle man which receives the information from one source and stores it with itself and then relays it to intended recipient when it is connected. Server can eavesdrop, manipulate or deny the message altogether. This dependency on server is risky for communication in organizations where security is of prime importance such as security agencies and scientific research facilities etc. In scenarios like Disaster Management and Rescue Operations deployment of server is difficult. Mobile devices have increased their potential by using powerful processors and enhanced storage capacity. The military uses a secure and captive network facilities exclusively, where this application can be deployed without any scope of misuse.

2 Application workflow

This application will be consisting of two broad components one for sending a file to another device and other for receiving the incoming file and saving it in device. These are called a sender and receiver respectively, by analogy to Internet based applications, a sender is

ICIDSSD 2020, February 27-28, New Delhi, India

DOI 10.4108/eai.27-2-2020.2303212

alike a web browser or a client and a receiver is a server. Hence each application contains a server and a client these components works independently to complete their functions for example an application's client may be busy sending requests to another application, while its own server may be waiting for incoming connections or may not have been started yet.

A receiver end, the listener process executes inside its event loop, the only event it handles is request for incoming connection, when encountered by incoming request, listener creates a connection handler and transfers the request to it. while connection handler processes and completes the request in another thread the listener goes back in loop to wait for another request until its is terminated by its parent. [1,2] Connection handler creates a request handler, get the response ,write it back on the same connection and finally ends its life-cycle. At client side, the operation is very simple. User tries to connect with another device, once connection is successful, sender asks user to select a file to be transferred. If a valid file is selected sender then prepares to send the file to receiver in multiple small chunks. Once file is sent through completely sender waits for user's action to select another file or close the application. Refer to the follojng Figure 1.

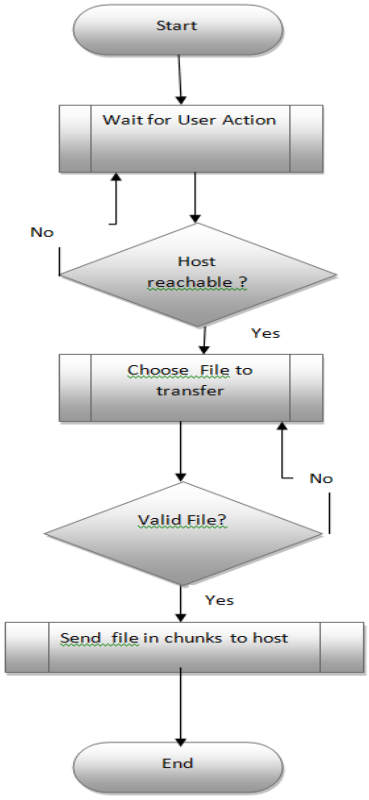

Fig. 1. Application Workflow

3 Design and Implementation

3.1 Server component

Server application is consists of tcp socket listener accepting incoming connections and processes the query (Request) and returns the result based on it (Response). Listener spawns a new instance of connection handler in a separate thread to handle each new connection so that multiple requests are accepted without blocking application. As the application is to be deployed on limited memory devices, there is a upper limit set for number of maximum simultaneous connections available.[3,4,5]. The requests and responses are managed by separate classes which follows paradigms of HTTP protocol. Once the response is written off the connection is released so that server can resume accepting new connections if it had paused due to application reaching maximum open connections.

3.2 Client Component

Client app is consists of a connection handler which tries to connect to http server. If initial connection attempt is not successful then Client shows Server not Available message and returns, if connection is successful, the client present the GUI to user to send appropriate file to server. The user selects the file through system‟s file open dialog. Once a file is chosen, client split the file into chunks and adds the file in process pipeline, pipeline is progressed when response of previously sent chunk is acknowledged by client. For each transfer the client uses a API KEY provided by server. Client asks for new API KEY before initiating a new transfer. Before transfering a chunk of file, client prepares a payload containing information about the command user intends to run on server for example Checking if server is up, requesting an API or sending a file chunk etc. The following diagram illustrates basic working of client app.

3.3 Listener Classes

This class is responsible for creating and initializing a tcp server which listens to the specified port of client machine. When a new connection is arrived, a callback is executed by system with the help of SIGNAL & SLOT mechanism of QT framework. This callback is provided with socket descriptor. Socket descriptor is used to identify the socket. Listener creates a new socket and assign to it the received socket descriptor. Listener callback then pass this socket to ConnectionHandler to start processing incoming connection. Listener extends the QTcpServer class provided by QT framework to provides signals for indicating new connection is arrived and to provides a listen function to start listening to a specified port.

The Request class HTTPRequest is responsible for processing the incoming request it works on a socket provided to it by ConnectionHandler through the Listener class. If there are bytes available for reading in a socket the connectionHandler calls the read methods of HTTPRequest class to extract headers and body of request. Once Request parsing is complete the connectionHandler calls HTTP Request Handler class to take appropriate action and

subsequently send a HTTPResponse. This whole process is one iteration of steps depicted in flowchart is illustrated in following figure 2.

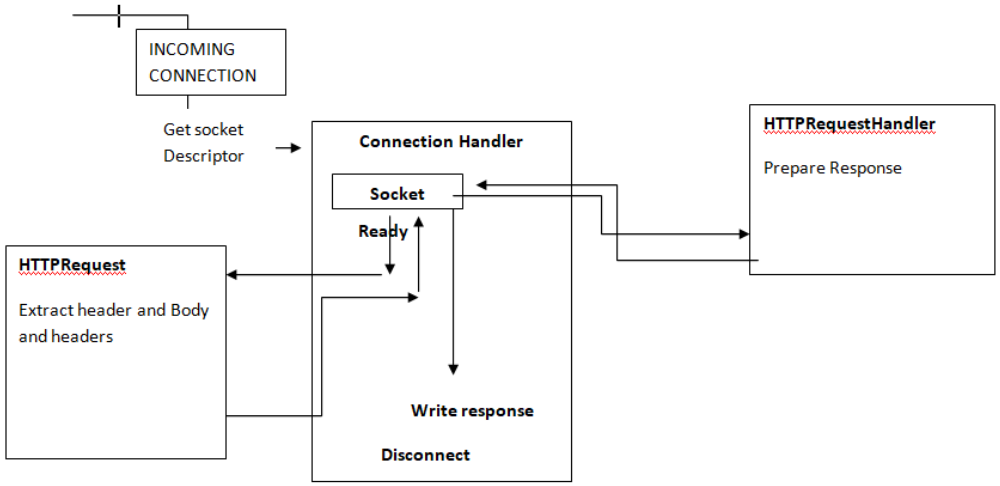

Fig. 2. Listener Cycle

4 Building and testing

4.1 Building APK with QT

Qt creator is an IDE that comes with QT framework supports the following methods for distributing the android application.

(i) As a standalone Package

(ii) As a minimal package containing dependency to Ministro tool. Ministro is a system wide Qt shared libraries installer/provider service. It acts as a bridge between apps and Qt libraries.

For this application it is not required to deploy on play store or generate a signed APK therefore we are using a single standalone apk generated by QT creator without any external dependencies of Ministro tool. Qt creator has android sources GUI to create and edit Manifest file of android application. Package name, Application Name, Minimum SDK , Target SDK and Permissions can be edited via from a GUI form. For Running application, can be done directly to a device or on a emulator. QT native ABI builder provides tools for managing android SDK and Android ABI. It can also detect ABIs created by AVD manager tool of Android studio. We have created ABI from Android studio as it provides latest downloads from official android sources. Android studio generates the apk with no dependency on system architecture, whereas in Qt creator deployment also requires configuration settings such as architecture of target machine such as X86 or Armeabi. Based on the architecture We have generated two apks for each model type.

4.2 Testing and Deployment

This application requires that device should have a capability of being contacted by another device or PC over Internet with the help of IP address. This IP address should be static or at least semi static so as to support file transfer with multiple requests in a session.
Due to the inherent IPv4 address shortage , any carrier uses IPv4 PAT (PORT ADDRESS TRANSLATION) to connect any subscriber to internet. Instead of assigning one address to each user, carrier uses one address for thousands of users. Carrier does this by assigning different port numbers to different users but uses single IP address for all of them. This limitation of 4g/3g Data connection make our application impossible for testing on mobile devices over conventional internet connection.Thus testing of applications is done on a closed network of static IP addresses.Refer to Table 1 below. Also military and department of defense issues the special broadband and internet services that provide static IP addresses to mobile devices over data connection. We have also arranged a setup to test our application on such environment for a very short duration. Due to the intrinsic security policy of department of defense and Indian army the application is tested for sharing some trivial files and images and was immediately uninstalled under the supervision of concerned officials. It was found that some executable files are deleted automatically upon completion of transfer on devices which have antiviruses installed.

Table 1. Results of the tests.

S.No	File Type	File Sizes (MB)	Result
1	Text(.txt)	1 - 50	Pass
2	Image(.jpg, .png, .gif)	1 - 50	Pass
3	Video(.mp4)	1 - 50	Pass
4	Compressed (.zip, .rar)	1 - 50	Pass
5	Executables(.exe)	1 - 50	Pass

4.3 Issues faced in testing and their workarounds

Mobile screen layouts and and widgets: One of the main problem with QT android development is that the look and feel of application on mobile devices. When Deployed on mobile screen the sizes of widgets and their positions within window changes completely . Android Studio solves the problem by maintaining size and positions in way that is neutral to device‟s screen size. It does so by specifying the metrics in density pixels instead of pixels. In Qt there is no support of Density pixels as for now, the solution to this problem is to use SVG images and layouts which scales in similar fashion on all device screens .The variation is minimal. Dialogs and popups: Some application requires popup and dialog services such as opening a file from file open dialog or opening the popup to show error message. QT does not provide native look and feel of such services on android devices. The appearance of these dialogs and popups exactly like in desktop application hence making it very difficult to use. Thus we have implemented our own popop and dialog managers to solve this problem. It is not as rich as native android but easy to use. Permissions : Android has changed its permission model to ask for permission only at the runtime when the feature of application is used for the

first time. This is supported with API Level 6.0. This feature is implemented by calling JAVA/Android native code from c++. For devices which are below API level 6, a permission must be granted manually by going to settings->Applications->SecureClient ->Permissions. Encryption and Decryption,: We are using simple encryption developed in house instead of using standard AES based encryption as that requires a development of services as a separate standalone project which could not be completed in given time frame.

5 Conclusion

The evolution of smartphones from IBM‟s personal communicator Apple‟s IPhone X has changed the way the information is stores and shared among the devices. The emergence cloud enabled devices has also made the file sharing and information sharing experience very easy and seamless. With a click of button the pictures and documents can be shared to people participating in a conference over Google‟s office or members of a WhatsApp group. There has always been a concern regarding the security of the files being shared. As a part of this project we have developed a file sharing application for android devices where server itself is embedded in a mobile application. Due to rise in computing power of mobility devices it is possible to run a mini server application in mobile device itself. In this model we have created a TCP based listener which is allowed to use only few number of simultaneous connections. furthermore the task of server application is limited to accept and parse a small set of commands only. The client on the other hand is allowed to share a file in multiple consecutive requests. One client cannot send file while a another file sharing is already in progress. File transfer is paused if due to some reason the host cannot be communicated or error in transmission had occurred. User can resume the sharing manually when network is up and running again. We have tested the application on special network arrangements which allowed allocation of fixed non sharing static IP address to mobile devices. This feature was available with few commercial carriers but now being discontinued due to shortage of IPV4 address space. The framework used for developing overall application is QT C++ for android application which allowed me to leverage my c++ programming skills to develop android application without learning new skills for mobile application development.

References

[1] Kumar R, Rajalakshmi S. Mobile cloud computing: Stan-dard approach to protecting and securing of mobile cloud ecosystems. Proceedings of International Conference on Computer Sciences and Applications; 2013. p. 663–9

[2] C Uddin M, Memon J, Alsaqour R, Shah A, Rozan MZA. Mo-bile agent based multi-layer security framework for cloud data centers. Indian Journal of Science and Technology. 2015 Jun; 8(12):171–8.4.

[3] Rajathi A, Saravanan N. A survey on secure storage in cloud computing. Indian Journal of Science and Technol-ogy. 2013 Apr; 6(4):1–6.

[4] Mishra A, Jain R, Durresi A. Cloud computing: Network-ing and communication challenges. IEEE Communications Magazine. 2012; 50(9):24–5.

[5] Rajarajeswari S, Somasundaram K. Data confidentiality and privacy in cloud computing. Indian Journal of Science and Technology. 2016 Jan; 9(4):1–8.