REMOTE MONITORING AND CONTROLLING OF RF COMMUNICATION FOR A MOBILE DEVICE

by

Raviteja Gadipudi

A Thesis submitted to the faculty of

The College of Computer Science and Engineering

In Partial Fulfillment of the requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, Florida

December 2013

REMOTE MONITORING AND CONTROLLING OF RF COMMUNICATION FOR A

MOBILE DEVICE

By

Raviteja Gadipudi

This thesis was prepared under the direction of the candidate's thesis adviser, Dr. Agarwal, Department of Computer & Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the college of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree if Master of Science.

SUPERVISORY COMMITTEE:

Ankur Agarwal, Ph.D.
Thesis Advisor

Shihong Huang, Ph.D.

Daniel Raviv, Ph.D.

Nurgun Erdol, Ph.D.
Chair, Department of Computers &
Electrical Engineering and Computer Science

Mohammad Ilyas, Ph.D.
Dean, College of Engineering and Computer Science

Barry T. Rosson, Ph.D.
Dean, Graduate College

November 21, 2013
Date

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help of so many people in so many ways. I must give my high, respectful gratitude to my academic adviser, Associate professor Dr. Ankur Agarwal for guidance and supervision. I like to thank Vijay Bollineni for his inspiration and effort to explain things clearly and in a simple way helped me to achieve my goals. I would like to thank Dr. Ankur Agarwal for giving me the opportunity to work as Graduate Research Assistant on the SNMREC grant.

I would like to sincerely thank my committee members Dr. Shihong Huang and Dr. Daniel Raviv for their valuable comments, suggestions to the thesis.

My deepest gratitude goes to my parents Venkateswarlu and Lakshmi for their love and support throughout my life, and also to my sister, Swetha. My cousins Haritha, Venkat, Narendra and Deepthi, have been a great support.

I offer my regards and blessings to all those who supported me in any respect during the completion of my thesis especially my friends.

ABSTRACT

Author: Raviteja Gadipudi

Title: Remote Monitoring and Controlling of RF communication for a Mobile Device

Institution: Florida Atlantic University

Thesis advisor: Dr. Ankur Agarwal

Degree: Master of Sciences

Year: 2013

In recent years there has been dramatic growth in mobile devices and technologies. According to reports from comScore [1], 47% users in the United States (aged more than 13) are using a smartphones as their primary phone. Smartphone offers more advanced computing ability and connectivity than contemporary phones. In today's world, a user wants to keep their smartphones private, because of the personal information present in it. Among these users, some of them are minors. This thesis addresses the functionality to track/control the mobile activities of minors by their parents using mobile phones. As a parent they want to know, whom his/her child is talking to and for what they are accessing browser for. Cellular network companies are providing number blocking services from the carrier side, but those are monthly paid services.

In this thesis, we propose application architecture for remotely control the child phone and grant access to selected numbers for call and text.

We use the emerging Android mobile platform and Google nexus phones to implement and test the application. This architecture will help developers to make more innovative applications in future which helps parent to access child phone information. We performed a study and reported the result using the proposal.

REMOTE MONITORING AND CONTROLLING OF RF COMMUNICATION FOR A MOBILE DEVICE

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 MOBILE PHONE EVOLUTION

In recent years, mobile phones have a vast improvement in the telecommunication technology. It is becoming a daily fixture of modern life. A mobile phone will help you to communicate with people when they are not near. Today a mobile phone [2] offer more services then past, services such as telephone, Short Message Services (SMS), Multi Message Services (MMS), gaming, wireless (Limited Distance) technology for connectivity and data exchange between devices, cameras, music players, GPS and browsing. There has been great improvement in the data connectivity like Wi-Fi, 3G and 4G for mobile phones resulted to client server cloud architecture application ecosystem [3]. Smart phones are also accompanied with sensors to help the user to get accurate data about their activity. Sensors like NFC, Accelerometer, Compass, Proximity sensor, Wi Fi, Bluetooth, Cloud services etc. Today a Smart phone has a wide decision power to process the data with the help of rich software, cloud support.

With the emergence of mobile operating system and its lightweight push messaging service, such as Android operating system [4], Google Cloud Messaging [5] has created the ability and reliability to transfer data from server to end Application. GCM will alert application if there is any new data from the server.

This thesis introduces challenges in the design and implementation of reliable way of data delivering system and then proposes solutions to address these issues.

1.2 MOTIVATION

Information Technology (IT) evolution in the recent years has been able to grow the mobility of the end users. This, in turn, has changed end users expectations and experience in the recent years in terms of available services, which have applications to find their place in daily life. New generation mobile phones are considered as smartphones. Third party applications [6] played an important role in smart phone development. The operating systems like android, windows, IOS give access to resources, API's for developers to develop more creative applications for the smart phones. There are many call and text blocking application present in various operating system markets like iPhone OS, android OS and lightweight push notifications in windows phone 8 OS (Microsoft Push Notification Service) [7], android OS (Google Cloud Message) and iPhone OS (Apple Push Notification Service) [8]. We adopted Android because it is an open source software stack, and a project led by Google, INC.

In today's world, we see there are lots of scams involve mobile phones [9] [10]. It's tough for the parents to keep track on child activity with phone every day. A mobile phone is a necessity for most adults. Parents are learning that when their child has a cell phone, it enables them to be safe at all times. But sometimes, children aren't necessarily mature enough to properly use such a powerful tool [11]. The main goal of this thesis is

to present the design and development of a remote accessing solution to block a call, text and data at child's phone. Here we are referring child phone as smart phone used by kids under parental guidance. AIM is to develop a mobile application which remotely, virtually controls the child phone. This goal is met if we have an application running on a phone should remotely control multiple child phones. Initial literature study is done in the domain of technology and product quality. With help of literature study, a quality prototype is designed and implemented on android platform.

1.3 REQUIREMENTS

In application development, physical and functional requirements are needed to perform a particular design or process. It is a statement that identifies a necessary attribute, characteristic, quality of a system for it to have value and utility to a user. Various requirements are listed below.

- Register the devices with the application server.
- Connection to transmit data between parent and child and vice versa.
- Control the list of contacts that can be dialed/ received in child device from the parent's device.
- Control the incoming text messages on child device from the saved application contacts list.
- Block websites thru proxy server on mobile network.
- Update the data on mobile phones while reinstall the application.

- Check whether the child phone is connected or not and let parent know when uninstalled.

1.4 PROBLEM CONSTRAINTS

1.4.1 COMMUNICATION CONSTRAINTS

Initially the application server is planned to include in parent application, and designed to send data through text messages, a lossless communication. Cost factor matters for this way of communication. To make it cost free, text messaging is replaced with GCM service in the system design to send messages to phone through cloud server, which is a time to live message. Server will destroys these messages when timed out and makes it a false positive.

1.4.2 CALL REJECTION CONSTRAINTS

This application requires telephony service to block selected incoming and outgoing calls. But the telephony services are changed to operating system level from android V2.3. To overcome this issue, the application uses a java concept called Reflection [12], which gives application methods to inspect other unknown classes, fields and methods at runtime. Any delay/ failure on accessing properties at application run time will proceed to connect/receive the call.

1.4.3 SMS BLOCKING CONSTRAINTS

Android manifest is the heart of an application. OS will check the manifest to invoke the appropriate application when there is an activity. On receiving a SMS the system will check the SMS intent filter priority of the SMS application in all manifests, the application with highest priority will be invoked first.

1.5 OUTLINE

The rest of the thesis is structured as follows

- The second chapter is focused on the literature review of research done for application development.
- Chapter three is focused on the study and design of system architecture. Moreover, it outlines the different tools and protocols used in system design architecture.
- The fourth chapter is focused on work flow.
- Chapter five is focused on results, future enhancements and conclusion for the application.

CHAPTER 2: BACKGROUND

This chapter outlines the elaboration and the response to android platform with third party applications and application development. At first it begins with background about mobile platform innovation and how it emerged in to today's world. Secondly, the platforms features are presented and explained. In addition, platform structure and platform benefits, and Android platform are identified and discussed as well.

2.1 MOBILE PLATFORM INNOVATIONS

Technology has greatly contributed to substantially increase our economic and social prosperity in every way. Innovations in mobile technology have been a driving force in today's world. A smartphone features more advanced computing capabilities and connectivity then a cellular phone. Smart phone era started with combining cellular phone with personal digital assistant (PDA), later we seen smartphones with additional functionality like media players, digital cameras, video cameras, GPS navigation, etc. in to one unit. Current generation high end smartphones also includes high resolution touchscreens, near field communication, web browser etc. Many smartphones are embedded with sensors called Accelerometer, proximity sensor, compass, light sensor, gyroscope etc. which helps the smart phones in many ways like battery optimization,

Location detection, screen brightness adjustment, motion etc. The first hand held Mobile Phone from Motorola weights about 2.2 pounds. Today most advance smartphone weights around 100 grams [13].

The mobile platform provides a set of features and resources for the development of applications and services. Most of the applications are third party. Developers who create software for mobile/software platform are called third party developers. These platforms help developers to earn money by distributing their applications also known as third party applications to consumers. New platforms making their platforms as open as possible and it requires fewer changes to the existing software to operate in their platform. We know Microsoft's Windows 8 application can be easily ported too WP8, the upcoming mobile OS Ubuntu supports most of the android mobile applications with minor changes [14]. Applications and services are asset to the mobile software platforms to increase users. Resources like SDKs (Software Development kits) [15], APIs (Application Programming Interfaces) [16], etc. helped developer to develop application in the platforms virtual environment with emulator. Successful software companies such as Google, Microsoft, Apple and Nokia have all adopted the concept of mobile platform in the early ages of mobile platform development, by creating a platform from which they themself used to build the products. Today the R&D cost of innovation in order for a company to solely build products around its platform would be too costly and unable to generate a wide variety of products, to meet the consumer demand [17].

2.2 MOBILE SOFTWARE PLATFORMS: MOBILE OS

Mobile OS is the operation system which operates a smartphone or tablet. Smartphone software development relies on Mobile OS. In 1993, IBM developed the first smart phone called IBM Simon with a touchscreen, email and PDA. After that we see mobile OS from Microsoft, Nokia, Blackberry, Apple, Google etc. The operating systems run on smartphones are the fundamental part of software system. The OS which have been designed for smart devices should be energy efficient with memory management and optimizations.

2.2.1 SOFTWARE PLATFORM STRUCTURE

A Platform is a formation of shared structures, which is divided in to four categories [18] and can be adopted in development.

1. ***Components:*** Storage disks, development tools, design patterns, resources, circuit design etc. are considered as components. A software development kit, An application programming interface are two core components adopted in development practices.
2. ***Processes:*** the equipment used to make components and the associated production and development process and supply chain. In development practices, a verification process applied by a platform's framework on application or service developed by a third-party developer is a one process type.

3. ***Knowledge****: A*pplications and limitations, Models, Testing methodologies, and design standards. In software development practices, knowledge can have various forms such as development guides, tutorials, documentations, guidelines, human interface guidelines, wikis, standard libraries associated with programing.
4. ***People and relationships****:* teams, relationships among team members, relationships with a network of suppliers and third-parties, and relationship between the team and the larger organization

As this application is developed on Android platform from Google Inc., we will take more about Android. Let's start with platform features and befits.

2.3 ANDROID OPERATING SYSTEM

Android, a Linux-based operating system designed for current generation mobile devices such as smart phone and tablets. In 2007, an Open Handset Alliance is formed by Google and a group of other mobile OME which became a world leading operating system. Android activated one billion of mobile devices in more than 190 countries around the world [19]. It's the largest installed base of any mobile platform and growing fast and every day millions of user's power up their Android devices for the first time. Android gives developers a world-class platform for developing applications for Android users with an open marketplace. Android users download more than 1.5 billion apps and games from Google Play each month. Android is continuously pushing the boundaries of hardware and software forward to bring new capabilities to users and developers. For

developers, Android innovation lets you build powerful, differentiated applications that use the latest mobile technologies. Java programming language [20] is used to develop android and the java code is converted by Dalvik at runtime.

2.4 ANDROID ARCHITECTURE

Android system [21] is divided in to four layers that start with application layer as top layer followed by application framework layer, system layer and Linux runtime core layer. Linux kernel acts as an abstraction layer between the software and hardware. Core services of the android system such as security, memory and process management, power, network drivers, driver model is done at Linux kernel version 2.6 [22]. Like every other OS, Android includes Dalvik virtual machine and a set of core functional libraries, written in Java language.

Libraries run on top of the Linux, contains the code that provides the features of an Android. The graphic libraries like Open GL|ES, SGL are core graphics libraries. A surface manager for display management, SQLite provides database support; SSL is used for internet security. Multimedia experience is taken care by Media framework provided by PacketVideo and a browser engine for WebKit.

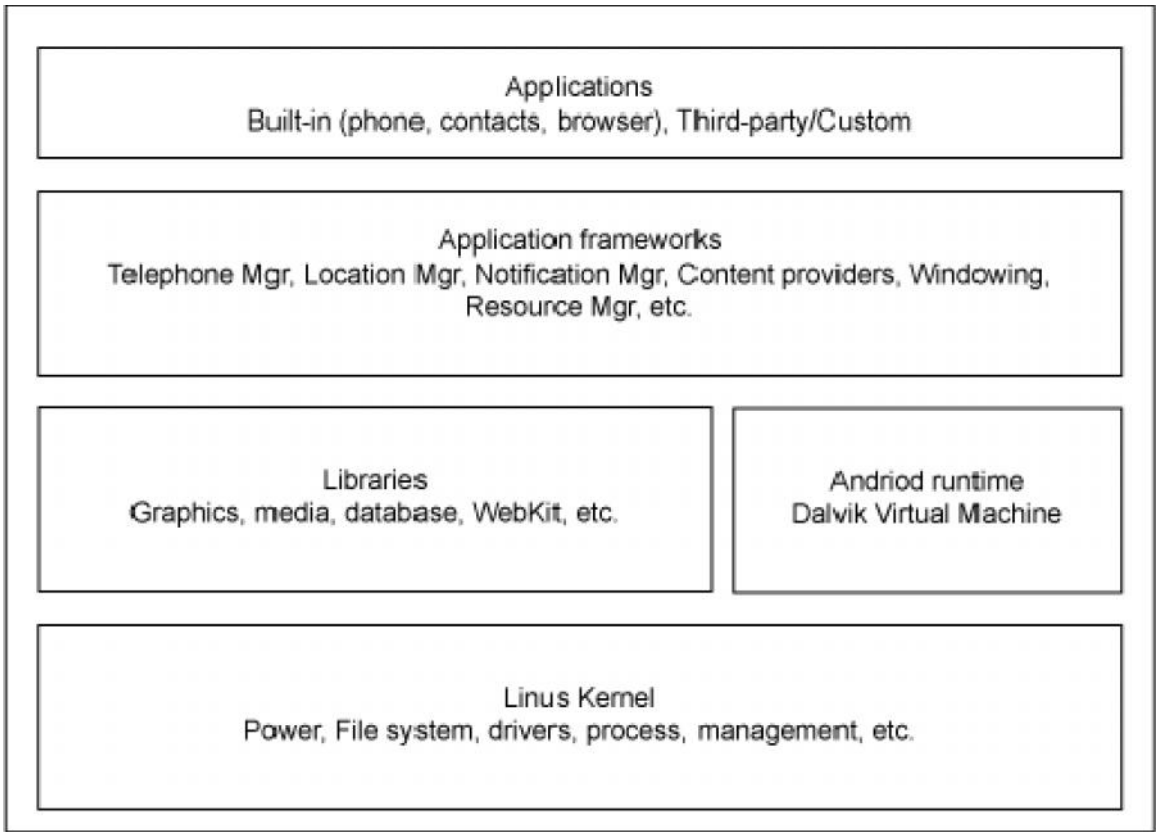

Figure 2-1 Android Architecture

Next is an Application Framework layer, which acts as toolkit for android operating system and applications. Applications are the top layer in the android architecture and it contains pre-installed and third party applications. These applications uses Application Framework layer to run.

2.4.1 ANDROID MANIFEST

Every application must have a manifest [23] file in its root directory. The manifest presents essential information about the application to the Android system, information

the system must have before it can run any of the application's code. Activities, services, broadcasts receivers, and content providers are the components of the application. These declarations in manifest let the Android system know what the components are and under what conditions they can be launched.

The android manifest does the following

- Manifest names the Java package foe the application, which serves as a unique identifier.
- It describes the components of the application.
- The permissions required for application to run on android platform are declared in manifest file.
- It lists the classes and libraries that the application must be linked.
- It declares the minimum level of the android API that the application requires.

2.5 GOOGLE SERVICES FOR ANDROID

Google offers a variety of services [24] that helps third party developer to build applications, help in building revenue streams, application distribution, and enhance application with features with maps, sign in and cloud messaging.

Google Maps: A developer can embed the map view in to the application. Map view can be customizable accordingly with markers, layovers, user perspective, and much more.

Google+ Sign In: A Developer no need to worry about maintaining the user login details. Google+ sign in feature in application allows the user to sign in with their Google account.

Google Cloud Messaging: Google Cloud messaging system notifies the users about the events by delivering lightweight messages from server, a free messaging service offered by Google.

Google Play In-App Billing: Built an application with a steady revenue stream that keeps users engaged by offering new content or virtual goods directly in your app. All transactions are handled by Google Play Store for a simple user experience.

Google Analytics: Google analytics gives developer the access to track in app purchases, number of active users, interaction patterns and much more.

Google AdMob Ads: AdMob offers the developers an alternative revenue opportunity that leverages multiple ad networks with targeted ads and several display formats.

2.5.1 ANDROID PLATFORM BENEFITS

Android has a low barrier to entry. There are no license fees for services and Google offers free development tools and resources for application development. Android applications are written in java, with a rich set of resources. It’s easy for developer with knowledge of java can get android application up and running. Android application can

be distributed thru many third-party application stores. The architecture and source code of the Android is open source with good platform documentation.

2.6 APPLICATION BACKGROUND

The earlier blocking application we have in the market is controlled by the user and also seen apps that can monitor device from other device by event injections. This application will allow parent to take care of child device, basically which gives a virtual access to selected services on child phone. In Application design view, Call handling and data transferring are important modules of the system. It’s also important for a parent to know the child phone is connected or not, Parent will have an option to check the child device status. Initially, parent needs to connect with child application to establish a trusted connection which needs to be done on child mobile. The application server will handle the connection information. Application custom contacts list reviewed by parent is saved locally on child phone, helps application to block when there is an unauthorized incoming call or text message. Phone with the child application installed can surf only to limited web sites on a mobile network.

Google cloud messaging service from Google for android OS phones played a key role in the system framework. GCM allows application server to send data from web server to users' Android-powered device. This is a lightweight message telling application there is new data to be fetched from the server (for instance, a request made by child), or it could be a message containing up to 4kb of payload data [25]. We see this service

mostly in IMs which can consume the message directly. This service handles all aspects of queuing of messages and delivery to the target Android application running on the target device. This application will generate a long-lived token, a simple API key which helps the server to send the message to target phone. Application server will maintain this key along with the phone number. When there is an event created on device, it will send the information to application server. The data received from devices is saved into server database and the data is packaged as a message. The application server will call GCM server to deliver message to device API key. On reinstallation, the application downloads related data saved on server into local device database.

The application manifest presents essential information about GCM message handling, call handling and text messaging to the Android system. When the android platform recognizes the activity related to the application, it directs the event to application with the help of manifest file and executes the code. If the event is incoming/outgoing call the platform will executes the application code and checks in local database, based on permission given by the parent it will block call without a ring. This application only blocks the incoming text messages based on parent permission.

2.7 RELATED WORK

Research is done on existing parental control applications available in market in market. Filtered four types of device control applications.

- Services offered by cellular network companies
- Applications that controls phones as a back ground service
- Applications installed as a launcher in phone
- Device to device viewer

2.7.1 CELLULAR NETWORK SERVICES

Major cellular companies are offering the parental control services from the network side [26]. This service runs irrelevant to user's phone and operating system. Americas GSM leader ATT offers smart control service in the name of, ATT wireless parental control is special paid service activated on user's interest. This is a monthly paid service, where user can block the bill purchases, calls, and texts. Call time and the data restrictions can be set. Verizon came with better parental control service. Like ATT, this is also a monthly paid service and the both services features device location tracking. User can manage Data, Voice and Message usage, voice services features call and spam blocking. The data service in Verizon is ahead from ATT, it offers content restrictions for different age groups. These services are not pure Remote Monitoring and Controlling of RF communication for a Mobile Device.

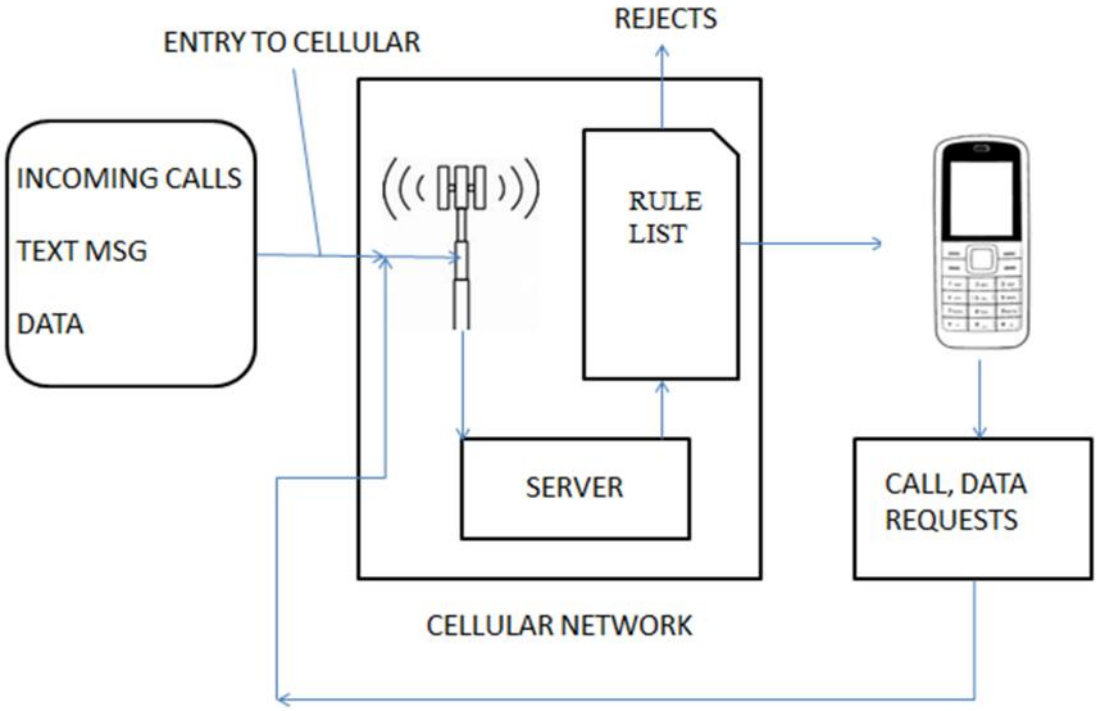

Figure 2-2 Architecture for parental control services (cellular based)

Above figure show the process of parental control at cellular network. All Calls, texts including data requests entered to cellular network will check the service subscription. The server validates the request with the rule list prepared by the user. The request failed to pass the rule list will be rejected by cellular service without notifying the user. For all this services the cellular companies provides a webpage to update the rule lists or by sending a text message. These are not pure device controlling services. Every time only parent can update the rule list. There is no way to let the parent know the child request when they are away.

2.7.2 OS BACKGROUND SERVCIES

PhoneSheriff [27] is a software/service which allows parent to monitor child's Android smartphone in real time. This software records and partially blocks the activities in child phone. PhoneSheriff application install into the operating system firmware which needs some special arrangements like rooting. It starts at every boot of the phone. After the software is setup on phone it will record SMS text messages, calls and other activities and then silently upload the data to your private PhoneSheriff account using the Mobile Internet. It also records GPS locations in intervals. When parent want to view activities list, they can view them on the child device or web browser and enter username and password to proceed.

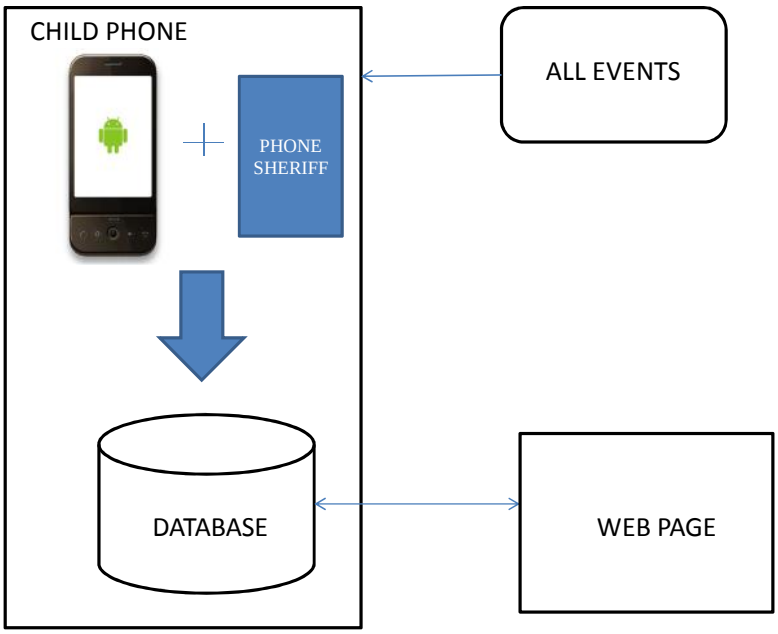

Figure 2-3 Architecture for OS based parental services

Above figure show the process of parental control with inbuilt software. This software logs the activity of child's phones and uploads in to database. It also gives the ability to filter content online. It records call, SMS, GPS and Data logs in the phone with the number/ content, data and time. This software also gives a feature to lock the phone for certain period. The call filter can be done only thru the website where parent can check the log. Like cellular parental control services, this is also paid service.

2.7.3 CHILD INTERFACE LAUNCHERS

Kytephone [28] is an android application for parental control. It turns an Android device into a child phone with parental controls. Kytephone creates a new home screen from which child cannot exit and allows parents to choose who their children can call and browse. With Kytephone's GPS tracking, parents can always locate their child from computer. It gives phone a child lock which allows parent to put the phone in kid mode. The application restriction will protect phone calls, block inappropriate websites. It acts as a call and SMS tracker. Parents can create a whitelist of contacts, allowing application to do call blocking and SMS tracking, so only approved contacts can contact your child. With device monitoring, parent can perform call logging, text logging, activity logging from Child application. Unlike previous parental control application, the Kytephone is free application can be downloaded from Google Play Store.

2.7.4 REMOTE DEVICE SURVEILLANCE

Android VNC [29] server application will let user connect to an android phone remotely via Wi-Fi or mobile network for control/view it. This is a basic device to device remote viewing application. But the application lacks the feature of parental control on child phone. It’s tough to view the activity in a remote phone all the time.

2.8 REVIEW

Above we had seen different type of parental and device to device control services/software. This thesis application will overcome issues like contacting the parent for new request, option to check the child device status; can grant/edit the permissions from the parent android phone, etc. The parent is virtually controlling the child device. This application is designed in the way that the child also will have some privacy from parent. The application can update/sync the data missed when it is offline or on reinstalling application.

	BLOCK CALLS/ SMS	MONITOR APPLICATION	LOG	PAID SEVICE	UPDATE CONTENT ON REINSTALL	CHECK CHILD DEVICE STATUS
CELLULAR SERVICE	YES	WEB PAGE	YES	YES	NA	NO
OS BACKGROUND SERVICES	YES	WEB PAGE	YES	YES	NA	YES
APPLICATION LAUNCHER	YES	NO	YES	NO	NO	NO
REMOTE DEVICE VIEWER	NO	PARTIAL	NO	NO	NO	YES
D2D REMOTE CONTROLLER	YES	ANDROID APP	NO	NO	YES	YES

Table 2-1 Comparing different parent control applications available in market

CHAPTER 3: SYSTEM DESIGN AND ARCHITECTURE

In this chapter, we present system design and architecture of the proposed Remote Monitoring and Controlling of RF communication for a Mobile Device.

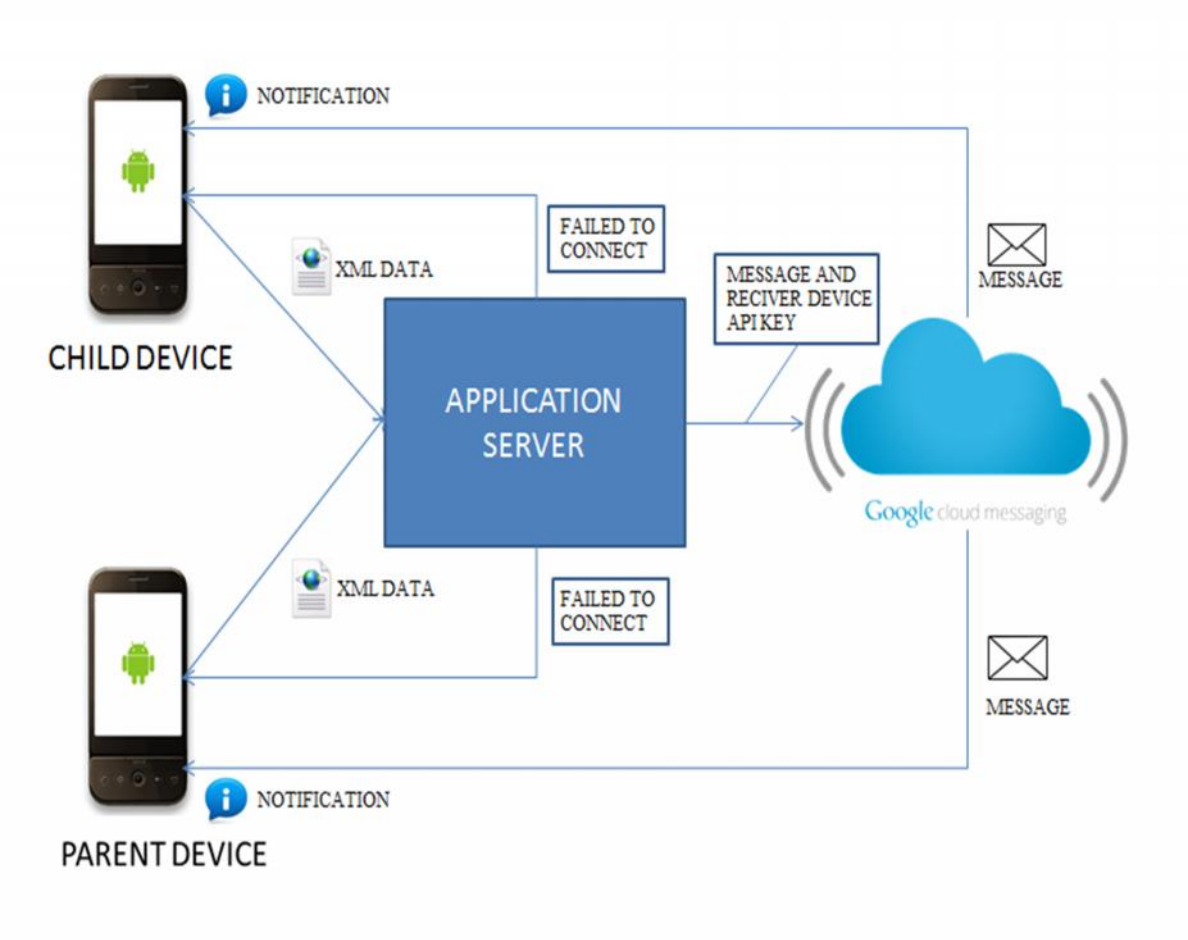

Figure 3-1 System Architecture

As shown in figure 3.1, the system comprises of a server that communicates the applications on devices. Remote Monitoring and Controlling of RF communication for a Mobile Device consists of two different applications (1) install on parent device, (2) child device. The components of the system are described in the following sub sections.

Mobile Client Components

- Android Platform
- Interface application
- SQLite database
- GCMBaseIntentService
- HTTP

Server Components

- J2EE Platform
- MySQL database
- GCM server
- Message builder
- HTTP

3.1 PRODUCT USAGE

This application requires smart phones with SIM and android as OS. This is deployed, debugged and tested on Samsung galaxy nexus phone and a Lenovo ThinkPad is used to run web service on server side.

Server side

- Intel i5 2.17 Gen 3 GHz with 4 GB RAM.
- Windows 7 professional.
- Sun Java Development Kit 6.
- Eclipse J2EE JUNO.
- Android SDK 4.0.

Mobile Side

- TI OMAP4 1.2 GHz processor
- 4G HSPA+
- Android 4.1

3.2 ANDROID APPLICATION

Android is the prevalent mobile framework with a rich ecosystem for application developers. We implement the mobile application on the Android Platform. The application front end used XML layout from the Android APIs which has neat and simple user interface. All activities done by the user can be done with it.

3.2.1 CHILD DEVICE

We developed a device control application for child phone to control selected services like call, text. All the calls, incoming text messages and websites are processed by the

application. The child application has the most important task to block the services which are not permitted by parent. The child application will register with GCM server and the generated long-lived token will be saved in application server for future communication. Application blocks all the calls and incoming texts initially. As first step, child device needs to setup a connection with parent device. Once the connection is successful, the child can request the parent for specific number. The response from the parent will be displayed as a notification. The application gives access to make a call by taping on number from the application itself. Child can view and recall the pending requests. A log is made to note the call history. The local database can also be updated manually.

3.2.2 PARENT DEVICE

An Android device with parent application installed is referred as a parent device. Parent application receivers the requests sent from the child application. Like child app, a long-lived token is created and saved in application server. A parent can add multiple child devices, where the parent will initiate the connection with child. Also, every request sent by the child is notified to parent and it will stores request details in the server and local database. Parent will have an option to edit the previous requests. The most important factor for a parent is to know whether the child device is connected to the network or not. The option “check child” in the parent application gives an option to know the child status. Parent application is also given option to save local database manually from server.

3.3 XML DATA FORMAT

XML is an extensible language [30], can create your own tags, or use the tags which have already been created. XML is used to send data to server. Its design is used to describe the data. The XML data is parsed by the xml parser presented in the application server. The parsed data is saved in to database as a data entry. SAXBuilder [31] is used for XML parsing. It's an event based sequential access parser API developed by the XML-DEV. It provides the data reading mechanism from XML documents.

3.4 APPLICATION SERVER

The System design is implemented using client server application architecture [32]. Client (Android application) builds the data as a XML and sends it to the application server over Hypertext Transfer Protocol (HTTP) [33]. The parsing and message building information is done on server side. The server communicates the result to the android application as a message and will be delivered thru GCM server. Application server is a web component run on a J2EE platform [34]. The J2EE technology is platform independent and has a rich set of application framework libraries. The server hosts the information relevant to the application on a MySQL database. The HTTP protocol is part of the system design. It provides a standard mechanism for data exchange between the server and the android application. Android also provides API to be invoked for this protocol. On incoming data, server checks for the data format to be XML. It is responsible for implementing the whole process of parsing, message building and

invoking GCM server. Server will notify device on communication failure. Database stores the parsed data from the server.

For application server, the JBoss application server [35], a JAVA EE certified is chosen for developing and deploying the java and web application. JBoss also provides the full range of java EE features [36].

3.4.1 MYSQL DATABASE

The database on server side carries the relevant information about the device address, established connection and transaction happened between parent and child. The schema for the database will depend on the target application type. However one common table set would be the request and response mapping between the applications. Database saves the device information when it registered with server. It also saves the time log for request and response. In Figure we show the server database schema.

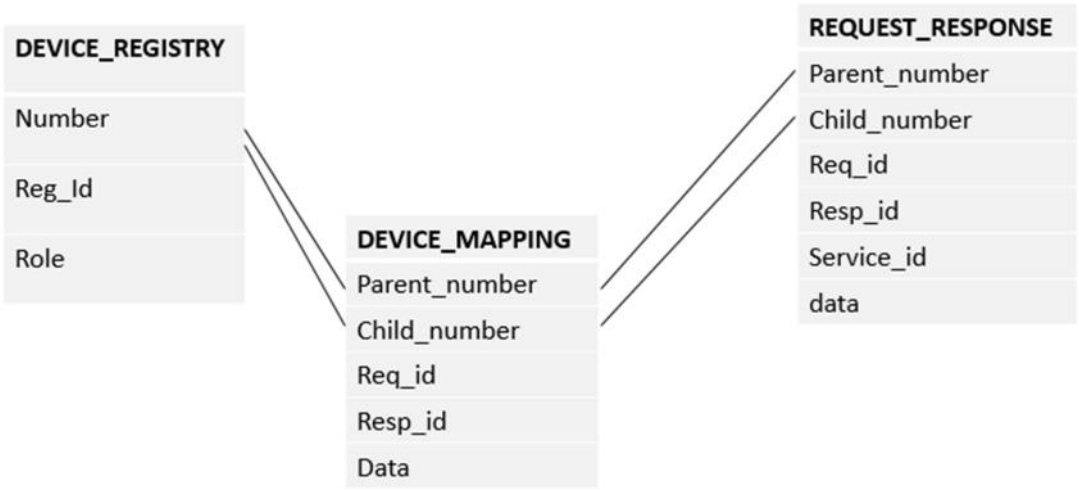

Figure 3-2 Database Schema

3.5 MESSAGE BUILDER

The server gathers the incoming data and builds a Message which is compatible for Google cloud messaging server. Instances of message class are immutable and should be created using a Message.Builder, class imports com.google.android.gcm.server.message [37].

```
Message message = new Message.Builder()
    .collapseKey(collapseKey)
    .timeToLive(3)
    .delayWhileIdle(true)
    .dryRun(true)
    .restrictedPackageName(restrictedPackageName)
    .addData("key1", "value1")
    .addData("key2", "value2")
    .build();
```

Figure 3-3 Message syntax with attributes and payload data

Messages are throttled on per application and per collapse key basis. Each application collapse key is granted some initial tokens, and new tokens are granted periodically thereafter. Each token is valid for a single message sent to the device. If an application collapse key exhausts its supply of available tokens, new messages are buffered in a pending queue until new tokens become available at the time of the periodic grant. Thus throttling in between periodic grant intervals may add to the latency of message delivery for an application collapse key that sends a large number of messages within a short period of time. Messages in the pending queue of an application collapse key may be delivered before the time of the next periodic grant, if they are piggybacked with messages belonging to a non-throttled category by GCM for network and battery efficiency reasons.

Every message posted my server to GCM returns a message ID. ID does not mean the successful delivery to the device. Rather, it means that it was accepted for delivery.

Factors after message are accepted by GCM server:

- If the device is ON and connected to GCM, the message will be delivered right away without any restrictions.
- If the device is connected but idle, the message will still be delivered unless the delay_while_idle is set to true. On other case the message will be stored in the GCM server until the device is awake. And that's where the collapse_key flag plays a role: if there is already a message with the same collapse key stored and waiting for delivery, the old message will be discarded and the new message will take its place. However, if the collapse key is not set, both the new and old messages are stored for future delivery.
- If the device is not connected to GCM, the message will be stored until a connection is established. When a connection is established, GCM will deliver all pending messages to the device, regardless of the delay_while_idle flag. If the device never gets connected again, the message will eventually time out and be discarded from GCM storage. The default timeout is 4 weeks, unless the time_to_live flag is set.

The restrictionpackagename flag will give developer to set the restrictions to particular application package name. The method addData will hold the payload data. Finally the build method builds the message in to one object.

3.6 GOOGLE CLOUD MESSAGING

GCM is a service that sends data from servers to android applications. GCM can deliver a message with 4KB of payload data. The GCM handles all aspects of queuing and delivering message to target android application on android device. Android application doesn't need to be running to receive messages. Intent broadcast will wake up the application when there is a new message arrived. The application need to be setup with the proper broadcast receiver and permissions. GCM passes raw message built in server. Developer can set notification, custom UI or silent data sync for incoming data.

The GCM server involves in taking messages from the application server and sending them to the device. It must have at least one logged in Google account if the device is running lower than API 15. Registration ID will be generated when the device register with the GCM, which is used by the application server to send the messages. Every android application is identified by the package name called application id from the manifest. The application id is helpful to validate the targeted android application. All the device registration IDs generated by the GCM server will be saved under sender id. The application server uses sender id to notify GCM server to identify device that has registered to. An API key that is saved on the application server that gives the application

server authorized access to Google services. The API key is included in the header of POST requests that send messages.

3.6.1 ENABLING GCM

To use GCM service in an application, it needs to enable the GCM. On first time the application fires registration intent to GCM server, this includes the sender id and application id. If the registration is successful, the server broadcasts an intent which gives the application a registration id. The registration ids are saved in application for future communication.

3.6.2 SENDING MESSAGE

Application server will build the message with payload data and retrieves the registration id from server database. The GCM server will find the device to deliver content with the help of registration id and API key.

3.6.3 RECEIVE MESSAGE

The system receives the incoming message and extracts the raw key/value pairs from the message payload, if any. The system passes the key/value pairs to the targeted Android application in Intent as a set of extras. The Android application extracts the raw data from the intent by key and processes the data.

3.7 NOTIFICATION

A notification is a message displayed to the user outside of application's UI. Application tells the system to issue a notification; it first appears as an icon in the notification area.

To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

CHAPTER 4: WORK FLOW AND CASE STUDY

4.1 OVERVIEW

We developed a sample application for implementing the framework described. This application provides the ability to recognize calls and text and blocking them in the mobile device. With this application, the user will be able to block the calls, text and data in the child phone and the child can request parent for a number to access call and text. A GCM server is created and implemented on server side to send the message created by server to device. All the devices with application installed are GCM clients. The information like device registration id, device phone number, request transactions are stored in the database. All the transactions are saved with request id and response id, this id will helpful for server to find the transaction information and device of origin. The mobile application is developed for Android OS version 4.0 (ICS) and up.

4.2 ANDROID APPLICATION DEVELOPMENT SETUP

Google Inc. provided developers a software development kit (SDK) to develop application for Android. This Android SDK has all the tools to build, deploy and test the application for android platform. An Android Virtual Device [38] (AVD) is an emulator Configuration comes along with Android SDK that lets developers model an actual

device by defining hardware and software options to be emulated by the Android Emulator. Using the emulator, the application can be fully tested before deploying on the real device. Eclipse is a multi-language software development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins. Eclipse supports Android SDK plugin to develop the application in integrated development environment. Eclipse also supports server side development.

4.2.1 GOOGLE CLOUD MESSAGING CLIENT SETUP

It's necessary to register a Google account with the Android Mobile Device to enable tools like Google Cloud Messaging on it. Application requires gcm.jar library to use the properties of GCM. Manifest needs custom permission and broadcast receiver shown below so only specific application can receive messages.

```
<permission android:name="my_app_package.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
<uses-permission android:name="my_app_package.permission.C2D_MESSAGE" />
```

Figure 4-1 GCM system Permissions in Manifest

```
<!-- App receives GCM messages. -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<!-- GCM connects to Google Services. -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- GCM requires a Google account. -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<!-- Keeps the processor from sleeping when a message is received. -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Figure 4-2 User permissions in manifest

```
<receiver android:name="com.google.android.gcm.GCMBroadcastReceiver"
android:permission="com.google.android.c2dm.permission.SEND" >
 <intent-filter>
  <action android:name="com.google.android.c2dm.intent.RECEIVE" />
  <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
  <category android:name="my_app_package" />
 </intent-filter>
</receiver>
```

Figure 4-3 Broad cast receiver code

This broadcast receiver is responsible for handling the intents that can be sent by GCM

- com.google.android.c2dm.intent.RECEIVE
- com.google.android.c2dm.intent.REGISTRATION

Should be defined in the manifest so that these intents can be received even if the application is not running. By setting the com.google.android.c2dm.permission.SEND

permission, you are ensuring that only intents sent by the GCM system framework are sent to the receiver.

<service android:name=".GCMIntentService" />

The above intent service will be called by the GCMBroadcastReceiver.

Every application uses GCM to receive message needs GCMIntentService class, which overrides the following methods.

- onRegistered(Context c, String regId): to register with server
- onUnregistered(Context c, String regId): to unregister with server
- onMessage (Context c, Intent i): called when an incoming message is detected my system.
- onError(Context c, String errId): called when GCM returned an error
- onRecoverableError(Context c, String errId): Called when GCM is unavailable.

```
GCMRegistrar.checkDevice(this);
GCMRegistrar.checkManifest(this);
final String regId = GCMRegistrar.getRegistrationId(this);
if (regId.equals("")) {
  GCMRegistrar.register(this, SENDER_ID);
} else {
  Log.v(TAG, "Already registered");
}
```

Figure 4-4 GCM registration Process

The checkDevice() method verifies that the device supports GCM. Similarly, the

checkManifest() method verifies that the application manifest contains meets all the requirements described above. Once the sanity checks are done, the device calls GCMRegsistrar.register() to register the device, passing the SENDER_ID you got when you signed up for GCM. But since the GCMRegistrar singleton keeps track of the registration ID upon the arrival of registration intents, we can call GCMRegistrar.getRegistrationId() first to check if the device is already registered.

4.3 SERVER IMPLEMENTATION

The server side of the application provides a feature to receive the incoming XML data and build a message to send to mobile device thru GCM server. It also supports a database that stores the device registration id and transaction details. The development of the server side components is done using the Eclipse J2EE IDE. JBOSS server is used to deploy and run the server. The implementation of the server side component is done using Google App Engine, which provides the client server architecture.

4.3.1 GOOGLE CLOUD MESSAGING SERVER SETUP

It's necessary to register a Google account with the Google API Console to enable Google Cloud Messaging and to obtain an API key from the API Console. In Google API console, a server key or browser key can be generated. We are using server key which allows the whitelist IP addresses. You can change the API key anytime from console. GCM server library is need to access and recognize GCM server from application server.

A servlet is created in the application server that can be used by the application to send the registration ID received by GCM. Server will uses the registration ID to send a message to the device. It uses the com.google.android.gcm.server.Sender class from the library to connect to GCM and send a message.

```
import com.google.android.gcm.server.*;

Sender sender = new Sender(myApiKey);
Message message = new Message.Builder().build();
MulticastResult result = sender.send(message, devices, 5);
```

Figure 4-5 Code to invoke GCM server to send data

The figure above does the following:

- Creates a Sender object using API key.
- Creates a message using a given registration ID (the message builder also has methods to set all message parameters such as the collapse key and payload data). Discussed in section
- Sends the message with a maximum of 5 retry attempts (in case the GCM servers are unavailable), and stores the response on result.

```
if (result.getMessageId() != null) {
 String canonicalRegId = result.getCanonicalRegistrationId();
 if (canonicalRegId != null) {
   // same device has more than on registration ID: update database
 }
} else {
 String error = result.getErrorCodeName();
 if (error.equals(Constants.ERROR_NOT_REGISTERED)) {
   // application has been removed from device - unregister database
 }
}
```

Figure 4-6 GCM error handler

4.4 APPLICATION FLOW CHART

4.4.1 APPLICATION REGISTRATION

Figure 4.7 shows the flow chart for registering child application and parent application. The user would initialize the application on both phones. On start, the application will check for a valid SIM and register with GCM. Once the application has the registration id, it will send it to server in XML format. The server will register with GCM to get the API key for future communication.

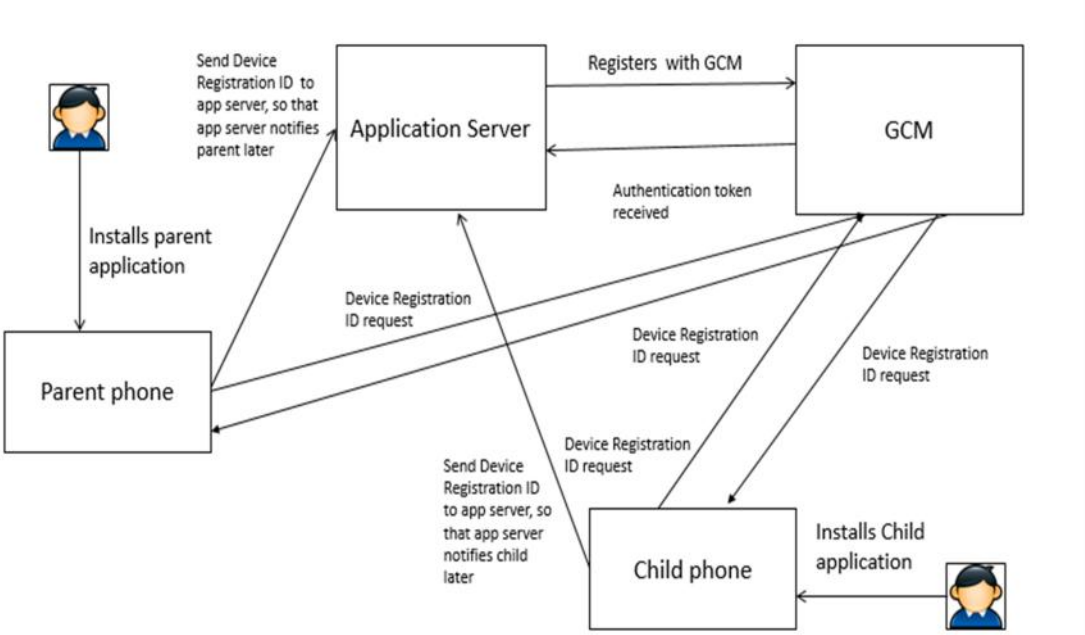

Figure 4-7 Device Registration to application server flow diagram

4.4.2 ADDING CHILD PHONE TO PARENT PHONE

Figure 4.8 shows the flow chart for paring the child application and parent application. The parent application will have the option to pair with child. The request is initially sent to server and application server will forward the request thru GCM and informs the child device. The paring will be successful with a positive response from child application. The child response sent directly to application server and it will maintain the details of all the parings in database. Finally, the process ends with informing parent's device about child status.

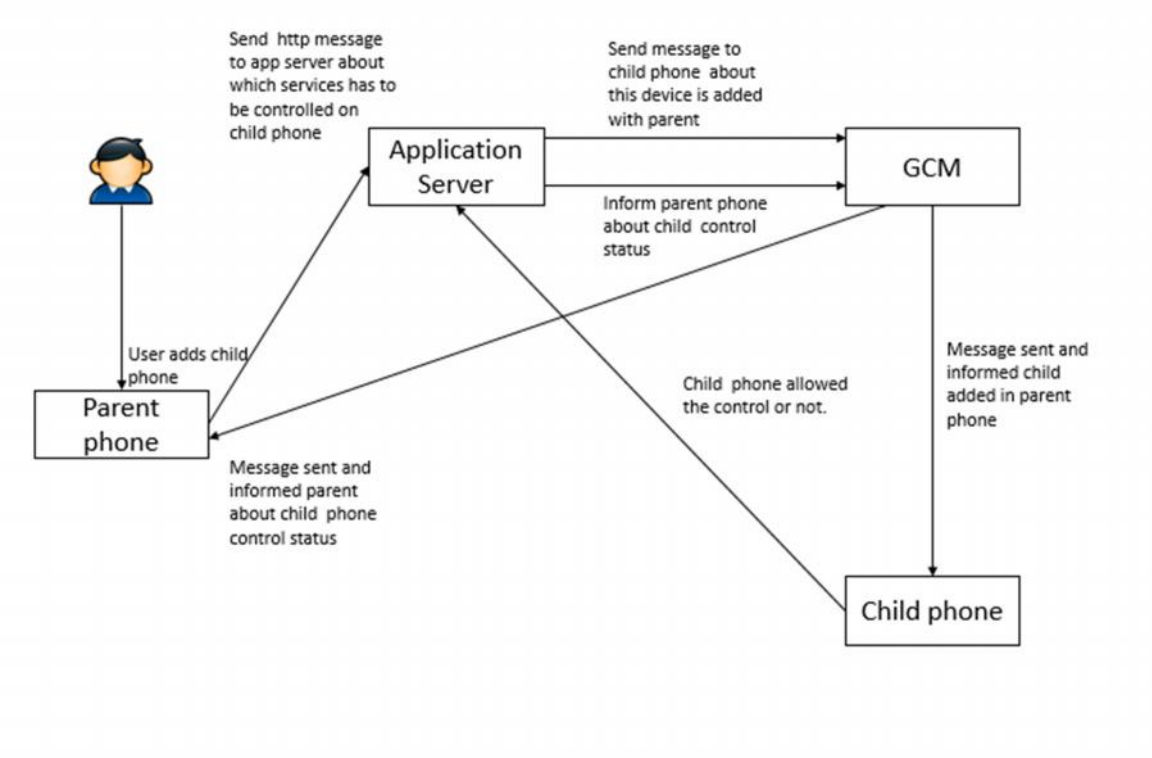

Figure 4-8 Adding child phone to parent phone flow diagram

4.4.3 REQUEST PARENT TO ACCESS CALL & TEXT FROM A NUMBER

Figure 4.9 shows the flow chart for requesting parent application for accessing call and text for a specific number. The child can only send the request to the paired parent numbers. The child request is sent to application server and it checks the database to validate the number. The request is forward to parent application and the application informs parent regarding the request from child. The parent response will be sent to application server to save in database and the GCM will informs the child application about the access.

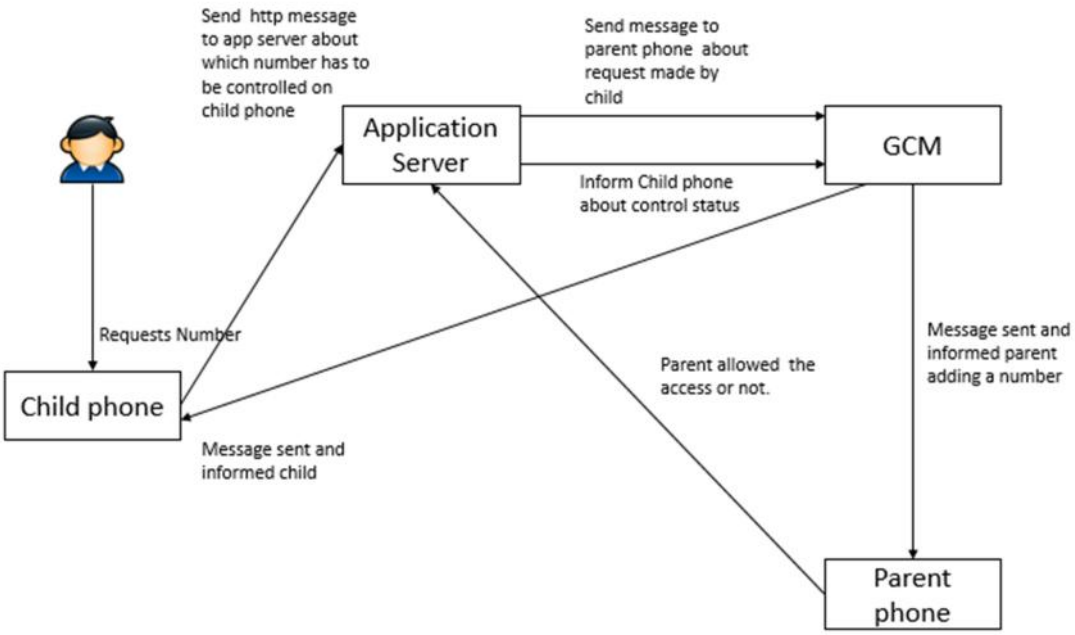

Figure 4-9 Child requesting parent to grant access to number flow diagram

4.4.4 UPDATE APPLICATION DATA FROM SERVER

Figure 4.10 shows the flow chart for updating application data on reinstall or sync message missed while offline. The user can update or sync the data on reinstall and missed while offline. The request from android application is sent to server and the server checks for the new data and sends it to the device thru GCM. The application server will also inform the android application once updated.

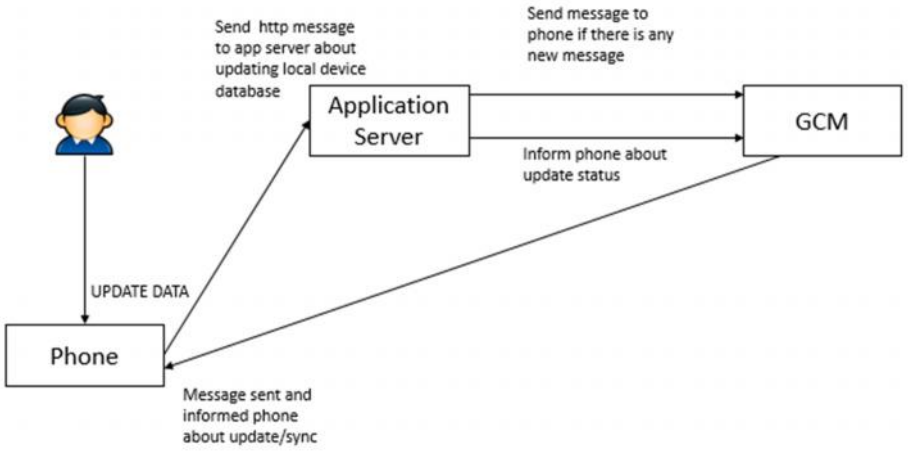

Figure 4-10 updating the application database flow diagram

4.4.5 CHECH CHILD CONNECTION

Figure 4.11 shows the flow chart for checking the child application status from parent phone. The parent can check whether the child is connected to server or not. The request from parent application is sent to server and the server checks for paired phones in databases and pings them. The response for the child phone to the application server will be informed to parent.

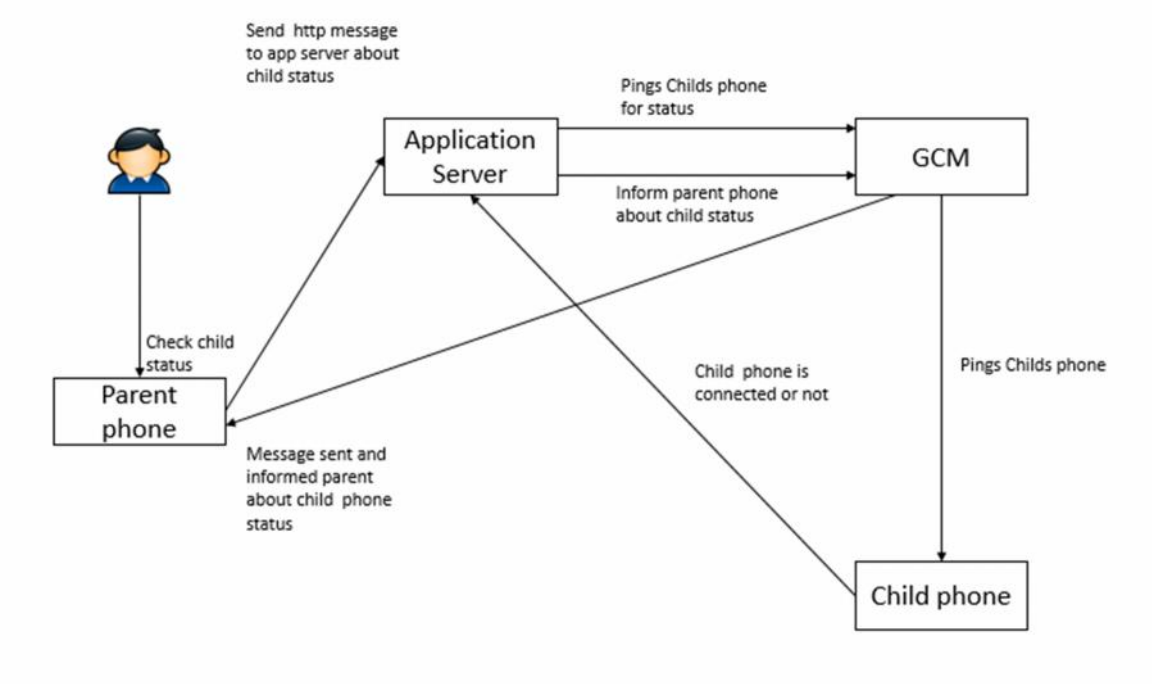

Figure 4-11 Check the Child status flow diagram

CHAPTER 5: SCREEN SHOTS, CONCLUSION & FUTURE WORK

Figure 5-1 Parent Application Welcome Screen

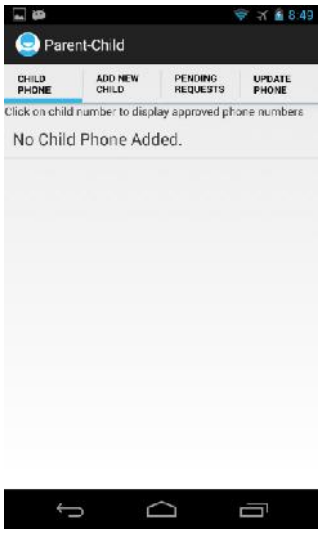

Figure 5-2 Main Screen Parent Application

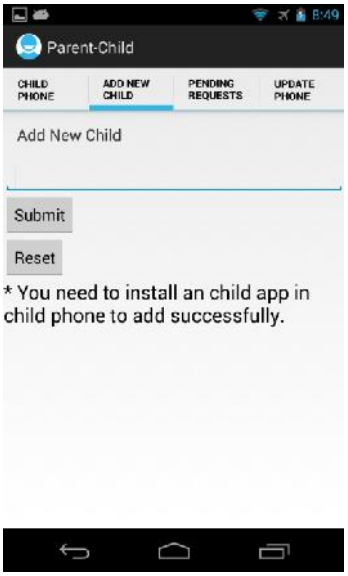

Figure 5-3 Add New Child

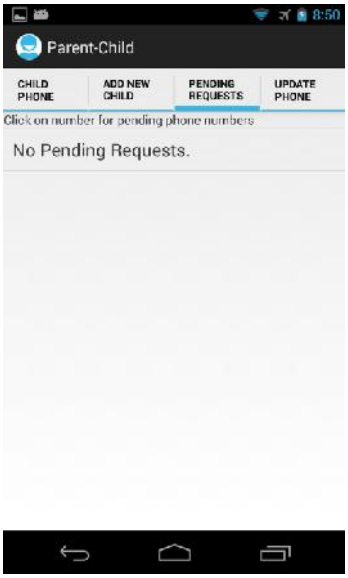

Figure 5-4 Pending Requests

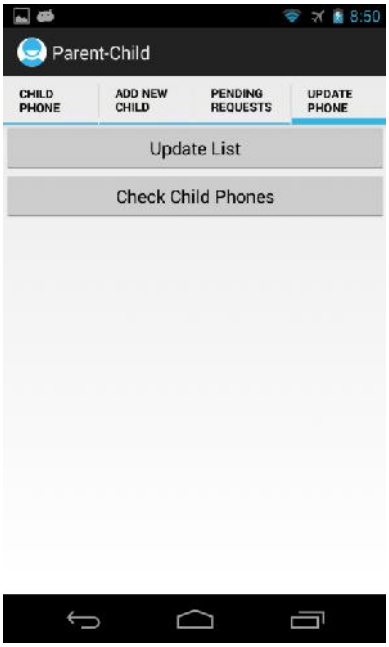

Figure 5-5 Update and Check Child

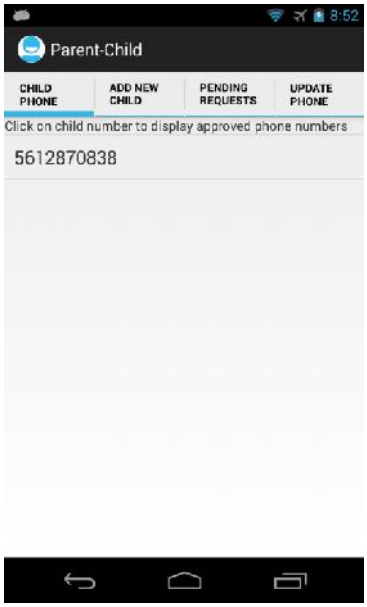

Figure 5-6 Child Added Confirmation

Figure 5-7 Toast Notification

Figure 5-8 Child GCM registration

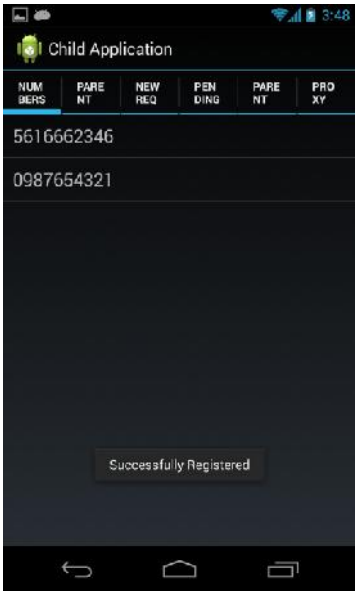

Figure 5-9 Server Connection Status

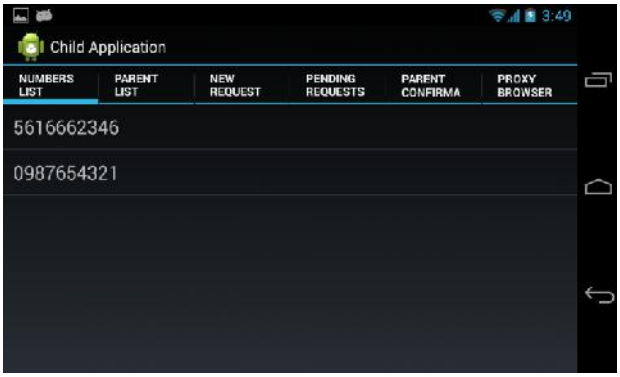

Figure 5-10 Main Page Activity

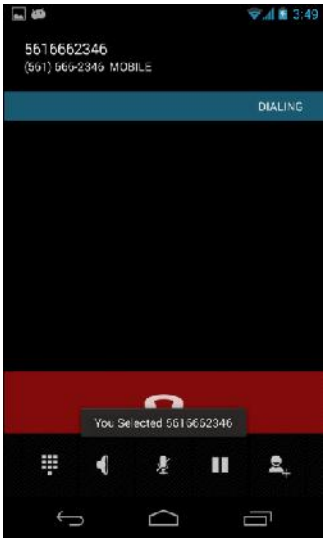

Figure 5-11 Call Activity

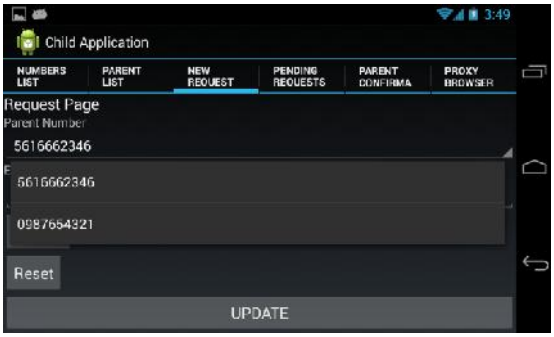

Figure 5-12New Request

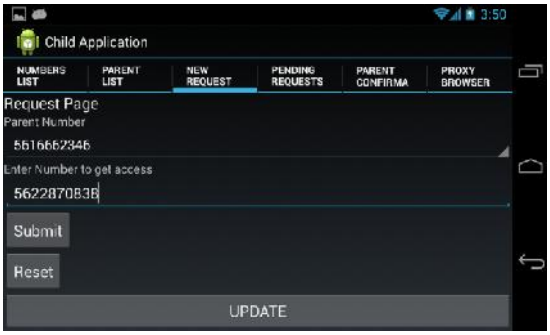

Figure 5-13 New Request Submission

Figure 5-14 Pending Requests

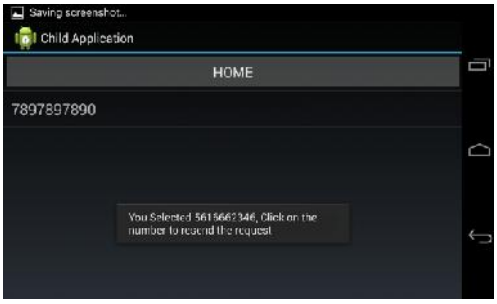

Figure 5-15 Resend Request

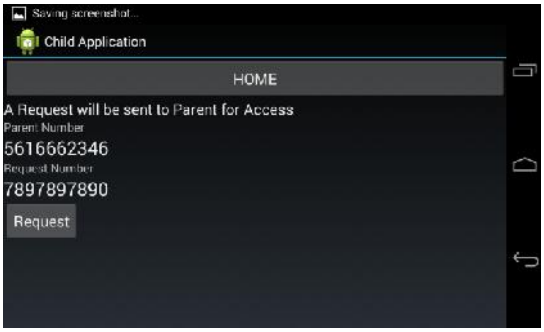

Figure 5-16 Request Confirmation

Figure 5-17 Parent Confirmation

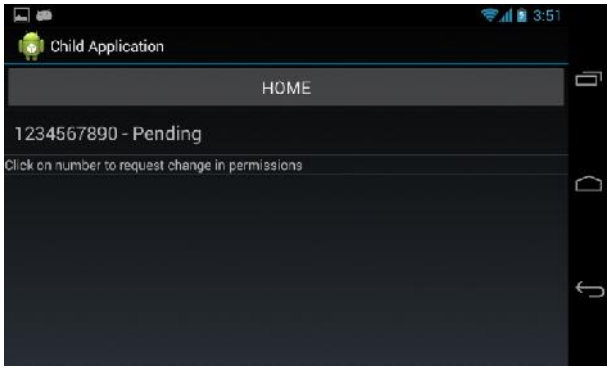

Figure 5-18 View Response

5.1 CONCLUSION

In This thesis, we proposed system architecture for Remote Monitoring and Controlling of RF communication for a Mobile Device on Android based mobile devices. The architecture is Client server model, which divides the effort between the mobile devices and application server for optimized communication. The architecture will contain two applications, specific for parent and child. The Child application will block the calls and incoming text messages, to get call and text access to specific number. Child phone needs to send request to parent application through application server using GCM. We proposed and implemented a robust virtual RF control on child phone with parent phone. We provided few third party application details which are used to control a device and populated a table with the features of each application. We also provided the design flow and screenshots of application for each module.

With this thesis we demonstrated a simple way to access the child phone remotely and control the phone via RF communication.

5.2 FUTURE WORK

The following avenues can be for further development of the work covered in thesis:

- Implementing GPS based all control blocking in child phone. Calling/ Texting and driving are in the news more and more these days. We here more accidents, state bans and statistics every day [39]. To help parent with their teen kids we can implement a GPS based call/text blocking, which prevents the use of phone RF Communication from child while driving.
- Automated access to parent on every module. Current application will let the child allow adding multiple parents. But the new features will grant the parent to get more control on child phone when adding a new parent number application requires permission from super parent. Also features like getting GPS location, forwarding keyword based outgoing messages and more.

REFERENCES

[1] "ComScore," [Online]. Available: http://www.newmediatrendwatch.com/markets-by-country/17-usa/855-mobile-devices.

[2] "Evolution of Mobile Phones.," [Online]. Available: http://www.hongkiat.com/blog/evolution-of-mobile-phones/.

[3] D. Chappell, "A Sort Introduction to Cloud Platform; an Enterprise - Oriented Overview," 2008. [Online].

[4] "Android," [Online]. Available: http://developer.android.com/about/index.html.

[5] "Google Cloud Messaging," [Online]. Available: http://developer.android.com/google/gcm/index.html.

[6] "Mobile Application," [Online]. Available: http://en.wikipedia.org/wiki/Mobile_app.

[7] "MPNS," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402558(v=vs.105).aspx.

[8] "APNS," Apple, [Online]. Available: https://developer.apple.com/notifications/.

[9] [Online]. Available: http://www.thejournal.ie/victims-wont-pay-for-slovenia-scam-772720-Jan2013/.

[10] [Online]. Available: http://www.thejournal.ie/phone-scam-lithuania-798200-Feb2013/.

[11] "AndroidSheriff," [Online]. Available: http://www.retinax.com/phonesheriff/uses.php.

[12] I. R. Forman, "Java Reflection In Action," MANNING.

[13] [Online]. Available: http://communities-dominate.blogs.com/brands/2009/11/celebrating-30-years-of-mobile-phones-thank-you-ntt-of-japan.html.

[14] "UBUNTU TOUCH," [Online]. Available: http://www.ubuntu.com/phone/ubuntu-for-android.

[15] [Online]. Available: http://en.wikipedia.org/wiki/Software_development_kit.

[16] "API," [Online]. Available: http://www.computerworld.com/s/article/43487/Application_Programming_Interface.

[17] M. Shoshah, "Third-Party Development Practices for Mobile Platforms".

[18] D. G. a. M. Shaw, "An Introduction to Software Architecture".

[19] Android. [Online]. Available: http://developer.android.com/about/index.html.

[20] K. Arnold, "The Java Programming Language," 1996.

[21] S. Brahler, "Analysis of the Android Architecture," 2010.

[22] "Linux Kernal 2.6," [Online]. Available: https://www.kernel.org/category/about.html.

[23] "Android Manifest," [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[24] "Google Services," [Online]. Available: http://developer.android.com/google/index.html.

[25] GCM Payload, [Online]. Available: http://developer.android.com/training/cloudsync/gcm.html.

[26] [Online]. Available: https://wbillpay.verizonwireless.com/vzw/nos/safeguards/safeguardLandingPage.action.

[27] "Phone Sheriff," [Online]. Available: http://www.phonesheriff.com/.

[28] "Kytephone," [Online]. Available: https://www.kytephone.com/.

[29] "Android VNC," [Online]. Available: https://code.google.com/p/android-vnc-

viewer/.

[30] "XML," [Online]. Available: http://www.w3.org/XML/.

[31] [Online]. Available: http://www.jdom.org/docs/apidocs.1.1/org/jdom/input/SAXBuilder.html.

[32] J. Dooley, "Software Development and Professional Practice," Apress, 2011.

[33] N. W. Group, "HTTP," http://tools.ietf.org/pdf/rfc2616.pdf.

[34] "Java EE at a Glance," [Online]. Available: http://www.oracle.com/technetwork/java/javaee/overview/index.html.

[35] "JBOSS," [Online]. Available: http://www.redhat.com/products/jbossenterprisemiddleware/application-platform/.

[36] A. Gupta, "J2EE Features," [Online]. Available: https://blogs.oracle.com/arungupta/entry/java_ee_7_key_features.

[37] "GCM," [Online]. Available: http://commondatastorage.googleapis.com/io2012/presentations/live%20to%20website/100.pdf.

[38] "AVD," [Online]. Available: http://developer.android.com/tools/devices/index.html.