

ROUTER REMOTE MONITORING USING  
ANDROID APPLICATION

CHONG JIAN DONG

BACHELOR OF COMPUTER SCIENCE  
(COMPUTER SYSTEMS & NETWORKING)

WITH HONOURS

UNIVERSITY MALAYSIA PAHANG



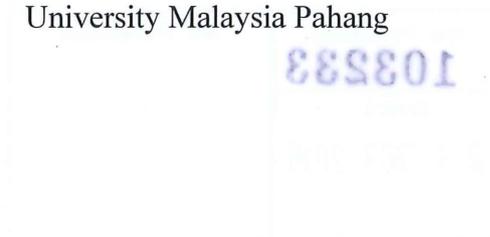
ROUTER REMOTE MONITORING USING ANDROID APPLICATION

CHONG JIAN DONG

A report in fulfilment of the requirement for the awards of Bachelor of Computer Science (Computer Systems & Networking) with Honours

Faculty of Systems Computer & Software Engineering

University Malaysia Pahang



DEC, 2014



UNIVERSITY MALAYSIA PAHANG

**BORANG PENGESAHAN STATUS TESIS**

JUDUL : ROUTER REMOTE MONITORING USING ANDROID APPLICATION

SESI PENGAJIAN : 2014/2015

Saya CHONG JIAN DONG

(HURUF BESAR)

mengaku membenarkan tesis (~~PSM/Sarjana/Doktor Falsafah~~)\* ini disimpan di Perpustakaan Universiti Malaysia Pahang dengan syarat-syarat kegunaan seperti berikut :

1. Tesis adalah hakmilik Universiti Malaysia Pahang
2. Perpustakaan Universiti Malaysia Pahang dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. \*\*Sila tandakan ( v )

SULIT (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD

Disahkan oleh

(TANDATANGAN PENULIS)

Alamat Tetap: 404A, KG BARU SUNGAI BATU,  
34900 PANTAI REMIS,  
PERAK.

(TANDATANGAN PENYELIA)

EN CHE YAHAYA BIN YAAKUB

Nama Penyelia

Tarikh: 5/1/2015

Tarikh: 5/1/2015

\*Sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis/laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

## STUDENT DECLARATION

I hereby declare the thesis “Router Remote Monitoring Using Android Application” is the result of my own research thesis except as cited in the references which have been used for acknowledged. This project has not been accepted for any degree and is not concurrently submitted for award of other degree.



---

(Signature)

Name: CHONG JIAN DONG

Matric Number: CA11039

Date: 5/1/2015

### SUPERVISOR DECLARATION

I hereby declare that I have read this thesis. In my opinion, this thesis is sufficient in terms of scope and standard for the submission of PSM, Bachelor in Computer Science (Computer System & Networking).

Signature :  \_\_\_\_\_

Supervisor : EN CHE YAHAYA BIN YAAKUB

Date : 5/1/2015

## **ACKNOWLEDGEMENT**

Firstly, I would like to thank my supervisor, En Che Yahaya Bin Yaakub for his invaluable advice and contributions to this project. His knowledge and high standards have definitely helped me to shape this work perfectly. It is my pleasure to have an advisor to being responsibility and caring to guide me in this PSM.

Last but not least, I am grateful to all my family members that sacrificed their time for me to make my work successfully developed. All my fellow friend and colleague should have my sincere appreciation for their support. Their tips and views are indeed very useful.

## ABSTRAK

Router Pemantauan menggunakan Android aplikasi adalah aplikasi mudah alih yang menyediakan fungsi untuk mendapat status router oleh pentadbir rangkaian dari jauh. Dalam persekitaran rangkaian, sentiasa memantau status router akan mengurangkan kerugian yang tidak perlu kepada satu organisasi apabila masalah rangkaian berlaku. Tujuan aplikasi ini adalah untuk menyediakan pemantauan mudah daripada peranti rangkaian router untuk pentadbir rangkaian. Dengan aplikasi ini, pentadbir rangkaian boleh mendapatkan maklumat rangkaian masa sebenar di hujung jari mereka di mana sahaja dengan syarat sambungan internet. Dengan sokongan penuh pemberitahuan menolak, pentadbir rangkaian akan tahu dengan serta-merta apabila terdapat kemas kini yang penting seperti interface gagal berfungsi, membantu anda menjejaki maklumat semua peranti dari tapak tangan anda. Oleh itu, pentadbir rangkaian boleh bertindak balas kepada masalah rangkaian secepat mungkin sebelum ia memberi kesan kepada peranti rangkaian yang lain. Kelebihan aplikasi ini ialah pentadbir rangkaian tidak akan terlepas apa-apa kegagalan rangkaian yang berlaku pada bila-bila masa. Aplikasi ini mempunyai beberapa had dari segi mengumpulkan maklumat, permohonan ini hanya melaporkan maklumat status router router seperti taraf interface. Masih perlu meningkatkan fungsi aplikasi ini seperti menambah laporan prestasi router.

## ABSTRACT

Router Monitoring using Android Application is a mobile application that provides the function to remotely viewing the router status by network administrator. In the networking environment, constantly monitor the router status will reduce unnecessary loss to one organization when network problem occur. The purpose of this development is to provide easy monitoring of the network router device for network administrator. With this application, network administrator can obtain real time network information at their fingertips from anywhere with prerequisite of internet connection. With full support of push notification, network administrator will know immediately when there are important updates such as interface down, helping you keep tab of all the devices from the palm of your hand. Therefore, network administrator can react to the network problem as soon as possible before it affects other network device. An advantage of this application is that network administrator will not miss any networks failures that occur at any time. This application has some limitation in term of information gathers, this application just report router status information of router such as interface status. There is still need improvement in order to increase the functionality of this application such as adding the performance report of the router.

## TABLE OF CONTENT

| CHAPTER  | TITLE                      | PAGE       |
|----------|----------------------------|------------|
|          | <b>Abstrak</b>             | <b>v</b>   |
|          | <b>Abstract</b>            | <b>vi</b>  |
|          | <b>Table of Content</b>    | <b>vii</b> |
|          | <b>List of Table</b>       | <b>x</b>   |
|          | <b>List of Figure</b>      | <b>xi</b>  |
| <br>     |                            |            |
| <b>1</b> | <b>Introduction</b>        | <b>1</b>   |
|          | 1.1 Introduction           | 1          |
|          | 1.2 Purpose of the Project | 2          |
|          | 1.3 Problem Statement      | 2          |
|          | 1.4 Objectives             | 3          |
|          | 1.5 Scope                  | 3          |
| <br>     |                            |            |
| <b>2</b> | <b>Literature Review</b>   | <b>4</b>   |
|          | 2.1 Introduction           | 4          |
|          | 2.2 Router Utility App     | 4          |
|          | 2.3 Router .CoCPit         | 6          |
|          | 2.4 PRTG for Android       | 7          |
|          | 2.5 Summary                | 9          |

|          |  |           |
|----------|--|-----------|
| <b>3</b> | <b>Methodology</b>   | <b>11</b> |
| 3.1      | Introduction   | 11        |
| 3.2      | Methodology for Router Monitoring Using<br>Android Application | 12        |
| 3.2.1    | Requirement Analysis   | 12        |
| 3.2.2    | System Design  | 12        |
| 3.2.3    | Implementation   | 12        |
| 3.2.4    | Testing  | 13        |
| 3.2.5    | Deployment   | 13        |
| 3.2.6    | Maintenance  | 13        |
| 3.3      | Overall Design of Application                                  | 14        |
| 3.4      | System Flow Diagram  | 15        |
| 3.5      | Database Design  | 20        |
| 3.6      | GUI Design   | 21        |
| 3.7      | Development Environment  | 25        |
| <b>4</b> | <b>Implementation</b>  | <b>26</b> |
| 4.1      | Introduction   | 26        |
| 4.2      | Database Development   | 26        |
| 4.3      | Application Development  | 27        |
| 4.3.1    | Application Server   | 27        |
| 4.3.2    | Mobile Application   | 32        |
| 4.4      | Testing Application  | 43        |

|          |                              |           |
|----------|------------------------------|-----------|
| <b>5</b> | <b>Result and Discussion</b> | <b>50</b> |
| 5.1      | Introduction                 | 50        |
| 5.2      | Advantages and Disadvantages | 51        |
| 5.2.1    | Advantages                   | 51        |
| 5.2.2    | Disadvantages                | 51        |
| 5.3      | Constraints                  | 52        |
| 5.4      | Summary                      | 52        |
| <b>6</b> | <b>Conclusion</b>            | <b>53</b> |
| 6.1      | Conclusion                   | 53        |
| 6.2      | Future Enhancement           | 54        |
|          | <b>References</b>            | <b>55</b> |
|          | <b>Appendix</b>              | <b>57</b> |

**LIST OF TABLES**

| <b>TABLE NO.</b> | <b>TITLE</b>                 | <b>PAGE</b> |
|------------------|------------------------------|-------------|
| 2.1              | Application Comparison       | 9           |
| 3.1              | Database table in App client | 20          |
| 3.2              | Hardware requirement         | 25          |
| 3.3              | Software requirement         | 25          |

**LIST OF FIGURES**

| <b>FIGURE NO.</b> | <b>TITLE</b>  | <b>PAGE</b> |
|-------------------|---|-------------|
| 2.1               | Add Device Menu                                     | 5           |
| 2.2               | New Device Setting Menu                             | 5           |
| 2.3               | System Flow of Router Utility                       | 5           |
| 2.4               | Router Searching Menu                               | 6           |
| 2.5               | Device Information Interface                        | 6           |
| 2.6               | Device Functions Menu                               | 8           |
| 2.7               | Device Account Setting                              | 8           |
| 2.8               | Concept of PRTG Android Connection                  | 8           |
| 3.1               | Overall Design of Application                       | 14          |
| 3.2               | Overall System Flowchart                            | 15          |
| 3.3               | App Client Register with GCM Server                 | 16          |
| 3.4               | App client Request Message with App Server          | 17          |
| 3.5               | App Server Send Message to App Client               | 18          |
| 3.6               | App Server Gathers Status Information from Server   | 19          |
| 3.7               | Main Menu of the Application                        | 21          |
| 3.8               | Interface to Retrieve History log Message           | 22          |
| 3.9               | Interface that show basic information of the router | 23          |
| 3.10              | Interface for Setting the App Server                | 24          |

|      |  |    |
|------|--|----|
| 4.1  | Mobile Application Database            | 26 |
| 4.2  | Application Server Debug Interface     | 27 |
| 4.3  | GCM Connection Code                    | 28 |
| 4.4  | GCM Connection Code                    | 29 |
| 4.5  | Handle Message Code                    | 30 |
| 4.6  | Send message Code                      | 31 |
| 4.7  | Create Data Message Code               | 31 |
| 4.8  | Main Menu Interface                    | 32 |
| 4.9  | GCM registration code                  | 33 |
| 4.10 | Send Registration ID to App Server     | 34 |
| 4.11 | Request Router Information Interface   | 35 |
| 4.12 | Handle Button Click Event              | 36 |
| 4.13 | Handle Intent Event                    | 37 |
| 4.14 | Create Notification Message            | 38 |
| 4.15 | Retrieve History Log message Interface | 39 |
| 4.16 | Handle Date Selection                  | 40 |
| 4.17 | Setting App Server Interface           | 41 |
| 4.18 | Handle Setting Button                  | 42 |
| 4.19 | Receive Notification                   | 43 |
| 4.20 | Display log Message                    | 43 |
| 4.21 | Display History Log Message            | 44 |
| 4.22 | Date with no Log Message Saved         | 44 |

|      |                                      |    |
|------|--------------------------------------|----|
| 4.23 | Reboot Request                       | 45 |
| 4.24 | Reboot Success Notification          | 45 |
| 4.25 | Request System up Time               | 46 |
| 4.26 | Display System up Time               | 46 |
| 4.27 | Request Person in Charge             | 47 |
| 4.28 | Display Person in Charge             | 47 |
| 4.29 | Setting Polling Interval             | 48 |
| 4.30 | Message receive after 1 minute       | 48 |
| 4.31 | Setting Bandwidth Usage Notification | 49 |
| 4.32 | Exceed Bandwidth Usage Notification  | 49 |

## CHAPTER 1

### INTRODUCTION

#### 1.1 Introduction

Router is a networking device that connects the data lines from different network to create an overlay of internetwork. It's located at any gateway where one network meets another network including each point of presence on the Internet. Its function is to forward the data packet between the computers network. A router may create and maintain the routing table of the available routes and their conditions and use this information along with distance and cost algorithms to calculate the best routes, and use this information to determine the path to direct the packets from one network to other network.

Monitoring is the use of a system to constantly monitor the computer network for slow and failing of the network components. When failure occurs, system will notify the network administrator via email or SMS. A network monitoring system help you to monitoring the network by generate all the network status and performance report, and this report will help you to understand bandwidth utilization and resource consumption. Monitoring the network can help to avoid any network problem that caused by undetected system failures.

Android is an operating system based on the Linux kernel, it's designed primary for touch screen mobiles devices such as smartphones and tablet computers. An android application is a software application that running on android platform. Android application commonly can be download from Google Play Store, where also known as android market. Android applications are primary written in the Java programming language and use Java core libraries.

## **1.2 Purpose Of The Project**

Router monitoring is an important network management criterion, constantly monitor the router status will reduce unnecessary loss to one organization when network problem occur. Therefore, monitor router status of the internetwork must be undergoing frequently. The failure of one link in router will eventually affect the entire routing process of the internetwork.

The purpose of this project is to develop an android application server which can obtain the router status in every time period, then the status of the router device will then be push to application client. This application will give the user instant insight into the device status such as bandwidth usage, events and link connection status. This development is aim to provide easy monitoring of the network router device for network administrator. With this application, network administrator can obtain real time network information at their fingertips from anywhere with prerequisite of internet connection. With full support of push notification, network administrator will know immediately when there are important updates such as interface down, helping you keep tab of all the devices from the palm of your hand.

## **1.3 Problem Statement**

In real networking environment, the operation of the router is critical for one organization, failure of networking device can bring to loss for one organization. Therefore, there is a need to constantly monitor the performance of networking device in order to react to the failure as soon as possible to minimize the loss for one organization. But monitoring the operation of router device will become difficult when network administrator is away from the workstation that generates the router performance report. So there will be a period that network administrator cannot react to the failure condition instantly. Therefore, the problem in the network which didn't handle right away will eventually affect other users or service in the network. Besides, troubleshooting the problems in the network will consume some time. This will eventually affect the overall network performance.

## 1.4 Objectives

There are three objectives for this project:

- i. To develop an application on the palm of the hand that can remotely view the status of the router.
- ii. Develop an application server to help network administrators to automatically generate the report of router status.
- iii. Alert the user about the network problem that occurs.

## 1.5 Scope

This application is mainly developed for network administrators as the user, help them to obtain the real-time alert information of the router device. And this application requires one computer performing the application server function to automatically obtain the information from the router. And this application server will send the status report obtained from the router to the application client by using Google Cloud Messaging service.

Plan to implement the scripting in the application server in order to automatically obtain the router status information. The application client will only act as a platform to view the status report sent by the application server, the application client doesn't have the ability to send the request or command from the client to the server.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

There are wide range of router monitoring tools are available to network administrator, basically have two type which are free and license version. By choosing the right router monitoring tool, you will be able to monitor router and troubleshoot problems before others network being affected. Below are three different kind of router monitoring application that I'm going to study and compare.

- a) Router Utility App
- b) Router .CoCPit
- c) PRTG for Android

#### 2.2 Router Utility App

Router Utility App is developed by Peplink Corporation, Peplink is the leader in internet load balancing and VPN bonding solutions. Router utility is the app using to monitor and control all the Peplink or Pepwave routers with push notifications status updates. But there is a limitation in this application, this app requires connectivity to a Peplink or Pepwave router and will not work without it. Router Utility application gives you instant insight into router device status, events bandwidth usage, and more. With full support of push notifications, you will know immediately whenever there is an important status update, helping you to keep tab of all your devices from the palm of your hand. With the router utility application it is easier to set up notification in the event your router gets rebooted or you have WAN failure. There are two options for push notifications, which are system startup and WAN up/ down. System startup notification is when there is a power outage, system reboot you will get a notification when your router comes back online. WAN up/down is the event when your WAN

goes down you will be notified that the link is no longer active, likewise you'll also get a notification when the WAN link comes back online.



Figure 2.1: Add Device Menu

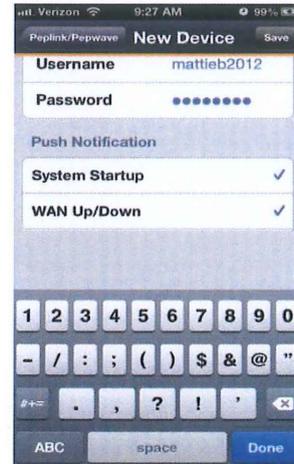


Figure 2.2: New Device Setting Menu

Figure 2.1 show the interface of initial startup screen of router utility where you can use to add router device. Figure 2.2 show interfaces where user need to enter all the relevant router login information such as hostname, username and password in order to properly adding new router device for monitoring. In figure 2.2, you'll notice that there are options for push notification that mention early just now. You have the choice to check or uncheck the options.



Figure 2.3: System Flow of Router Utility

Figure 2.3 show the system flow of router utility app by Peplink, app client have the ability to send command to router for login or acquire status information and receive notification from the router.

## 2.3 Router .CoCPit

Router .CoCPit is an android application develop by Marmiko IT-Solutions, this application let you control and monitor your UPnP capable router in real-time. With Router .CoCPit, all the essential performance and status data of your router will be displayed and constantly updated. UPnP stands for Universal Plug and Play, UPnP is a set of networking protocols that allow networking devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing and communications. Router .CoCPit only works with the router that support UPnP and it's active in the router. The Router .CoCPit include simple and easy to use interface, have clear overview with all important data constantly updated. This application has the function to automatically find one or more supported routers in your network instead of manually input the network devices IP address. By using this application in the network, detailed information such as upload and download speed, connection statistics and status, IP addresses and devices details is instantly available.



Figure 2.4: Router Searching Menu

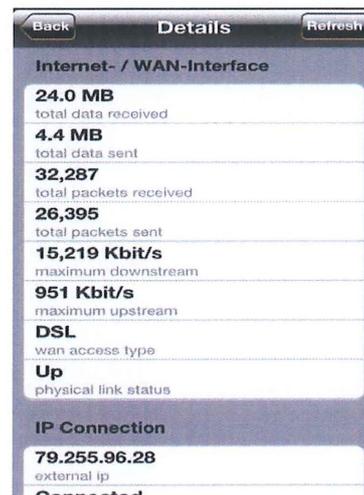


Figure 2.5: Device Information interface

The figure 2.4 interface above show the function to search for an UPnP enabled router. If success to find a router, application will be automatic connect the router and you will be able to view the status information of the router. Figure 2.5 show the interface where you can viewing the detail information of the connection.

## 2.4 PRTG for Android

PRTG for Android is a free Android app that connects your smartphone or tablet to your PRTG installation. It brings the power of the PRTG Network Monitor interface to your Android 4.0 or later devices, but it requires the installation of PRTG Network Monitor. PRTG Android lets you monitor the network remotely, where this application will connect to PRTG servers through HTTPS or HTTP over mobile, Wi-Fi or VPN networks to show network status. With this application, network administrators can obtain the network status and manage the events from wherever they are. Alarm notifications will display in the phone's notification area when network problems occur and users can use this application to view the live data.

PRTG Android is a clear and simple viewer for the entire network monitoring environment, where the application will request the monitoring data directly from the PRTG server and display it in the application. Therefore, the PRTG server must be reachable from the internet in order to use PRTG Android. The PRTG server, which is the core of the PRTG installation and includes data storage, report engine and a notification engine, and this core server is configured as a Windows service which runs permanently. The core server is the heart of the PRTG system and performs the configuration management for object monitoring, notification management including a mail server for email delivery and report generator and scheduler.

This application supports multiple accounts to query different server/login combinations, this can let you to switch easily between different accounts to quickly view monitoring data of several installations and/or accounts. Besides, this application can check all accounts and automatically in the configured polling interval and notify you in your device's status bar whenever there are alarms for one of them.



Figure 2.6: Device Functions Menu



Figure 2.7: Device Account Setting

Figures 2.6 show the PRTG android device functions menu, at here user can view the monitor device, setting alarms and ever showing the connection maps of the network. Besides, user also can view the reports that generate by the core server and also the logs being save in the core server. At figure 2.7 interface, user can edit the setting and viewing the detail information of this application.

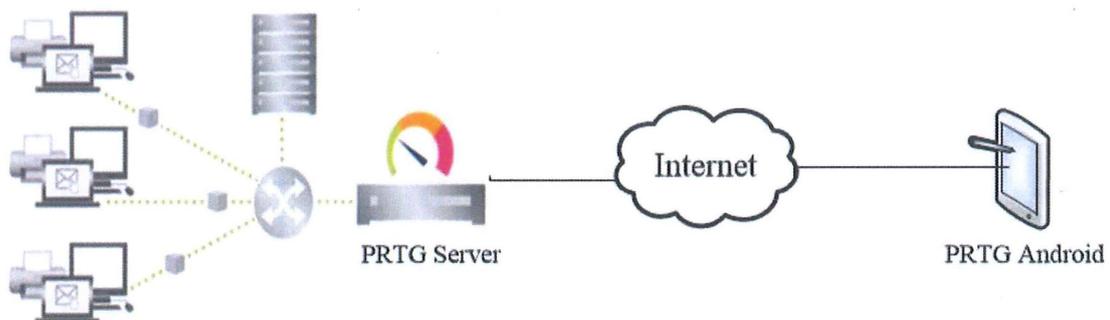


Figure 2.8: Concept of PRTG Android Connection

Figure 2.8 show the basic concept on the connection of the PRTG android to the PRTG server. Where PRTG will sit inside the organization network monitoring the router, and user must create an account in PRTG server in order to use by PRTG Android to connect remotely. Then PRTG android application will connect while local area network WI-FI or mobile internet, then user require to add the PRTG server account in the application setting in order to viewing the router status update send by the PRTG server.

## 2.5 Summary

There are a lot of monitoring tools that developed by corporation and individual but the tools that include the function remotely viewing the status of the router device is not more. The function of remotely monitoring the network devices can be critical when network administrator is away from the workstation and the network problem occurs at the same. Therefore, with application that can bring the real time status information of router to the palm of the network administrator will certainly decrease unnecessary downtime to the network.

From the finding, there are three android applications that can bring the network real time status to the palm of the network administrator. But certain application requires the application server side to be functioning before application client can obtain the real time status update such as PRTG android application. The problem for Router Utility application is that it cannot support other router brands besides the Peplink/ Pepwave router, so this will affect the usefulness of this application. Besides, Router Utility and Router .CoCPit application is limit for the use in local network where the connection between the application and network device must be in the same network in order to functioning. In these three applications, there is the strengths and weakness in each of them. In order to make a clear comparison of the existence application to the application that want to develop, table will be draw at below to show the clear comparison of each application.

Table 2.1: Application Comparison

|                       | Router Utility<br> | Router .CoCPit<br> | PRTG for Android<br> | Router Monitoring<br>Using Android<br>Application |
|-----------------------|---|---|--|---|
| Router device support | Peplink brand router  | Router that Support UPnP  | All routers brands   | All routers brands                                |
| Connectivity Range    | Local area network  | Local area network  | Local area network and external network  | Local area network and external network           |

|                            |                            |                            |                            |                            |
|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| Require account server     | No                         | No                         | Yes                        | No                         |
| Android Version            | Require android 2.2 and up | Require android 2.2 and up | Require android 4.0 and up | Require android 2.2 and up |
| Require application server | No                         | No                         | PRTG 13.1 or higher        | Yes                        |

From the table above, we can observe that strength of PRTG Android and our Application is that it can support wide variety of router device but Router Utility and Router CoCPit only supports some type of router device. For connectivity range, PRTG Android and our Application can connect to external internet to remotely viewing the status of the router device, but Router Utility and Router CoCPit only work inside the local area connection. The strength of the Router Utility and Router CoCPit is that they not need to install the server on the workstation in order to viewing the status of the router. The weakness of PRTG Android compare to other android application is that it only support android 4.0 and up device and require the installation of PRTG 13.1 and higher.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Introduction

System development methodology is the standard process followed to conduct all the steps necessary to analyze, design, implementation and maintain information systems. Methodology or called design description in this project is the framework that used to structure, plan and control the process of developing information systems. There are many type of methodology like as Systems Development Life Cycle, Spiral, V-Model and many.

Waterfall model is a sequential development pattern, which development is look as flowing downwards (like a waterfall) through the phases which are requirement analysis, system design, implementation, testing, deployment and maintenance. Analysis describes the current system and the limitation of the system with the opportunities to enhance the current system. Design specific the functional, detailed specifications of all systems elements including the data, process, inputs and outputs. Implementation involves the coding of the system and documentation. Testing is the entire system is tested for any faults and failures. Deployment is where the product is release in the market. Maintenance is where the issues come out in the client environment and updated system and documentation are release to enhance the product.

Spiral model is a software development process which combines design and prototyping-in-stage, in attempt to combine the top-down advantages and bottom-up concept. This model also focuses on the early identification and reduction of the project risks.

V-model is the extension from waterfall model which are based on association and testing phase for each or every matching development stage.

## **3.2 Methodology for Router Monitoring Using Android Application**

The methodology chosen and applied in this project is Waterfall Model. The waterfalls have six phrases which are requirement analysis, system design, implementation, testing deployment and maintenance.

### **3.2.1 Requirement Analysis**

In this phase, requirement that is needed for this project is gather and analysis. To know more about requirements and function of this project, study are needed on the current applications that have been developing in the market. From the study, require to pin point the weakness of the current application then create a solution for this problem in order to implement in our application.

### **3.2.2 System Design**

In the design phase, the conceptual design about system will be starts based on user requirements. Design that will be done on this phase is through drawing. System flow diagram, use case diagram, database design diagram and flowchart will be draw based on the requirements and functional. Besides that, the virtual or sample interface of the Router Monitoring Using Android Application also design and create based on the diagram drawing, this is to get the idea about the real interface design of system.

### **3.2.3 Implementation**

After having the design plan, diagrams and virtual interface, the system will start to code. In this phase, the system will be code by using the Java Language and Object Oriented Programming Language. The coding will be writing based on each functional part of the application, and then continue with integration part. It will continuous until the application is complete.

### **3.2.4 Testing**

After the application is complete produced, the each functional part of the application will be tests in order to local any error in the application and this error is fix before the entire application is put on the test. After each of the functional unit being tested, whole application will be run to make sure the application can be run properly and stable ability. After that, the application will be ready to deploy in the market.

### **3.2.5 Deployment**

After the testing of each functional unit is complete, the product will be ready to deploy in the customer environment. After that, the evaluation and feedback of the client will be obtained from the market and this feedback can be used to enhance our product in the future. Besides, this phase can help to obtain any issues or error which comes up in the client environment.

### **3.2.6 Maintenance**

In this phase, require to work on the issues or feedback that obtains from the client environment. Then the new version of the application is release to solve the issues and enhance the product. Maintenance is done to deliver these changes to customer environment.

### 3.3 Overall Design of Application

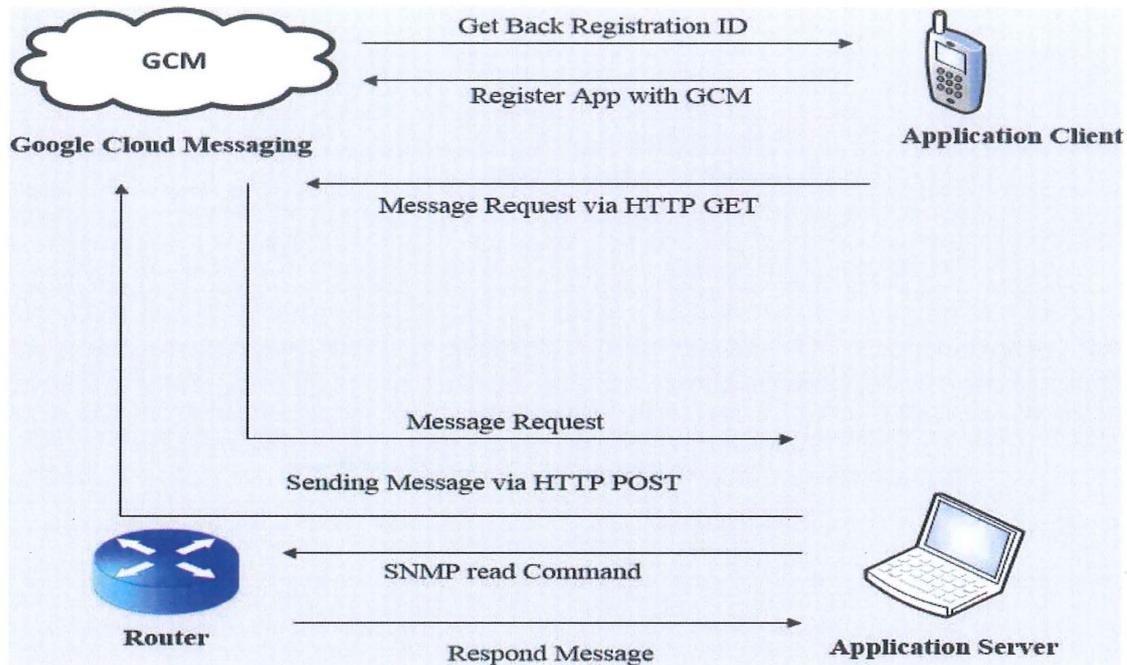


Figure 3.1: Overall Design of Application

In the figure 3.1, Google Cloud Messaging (GCM) mechanism plays an important role in order for application client to receive and request the router status from the application server. In order to use GCM, one API project needs to create and obtain the project ID and API key. Because the project ID and API key are needed in order to communicate with GCM. Next, require to develop the android application and application server to communicate with the GCM. In the code of the android application need to include the project ID that allow the application to register with GCM. API key obtained also needed to include in the code of the application server in order to use as the password to authenticate with the GCM Server.

After that, continue to develop the java application that can run in the application server to acquire the status of the router. Java application will be develop using the 3<sup>rd</sup> party library which is snmp4j to communicate with the router. Then, the information obtain from the router will be save in the database and server will send the information in the database to application client.

### 3.4 System Flow Chart

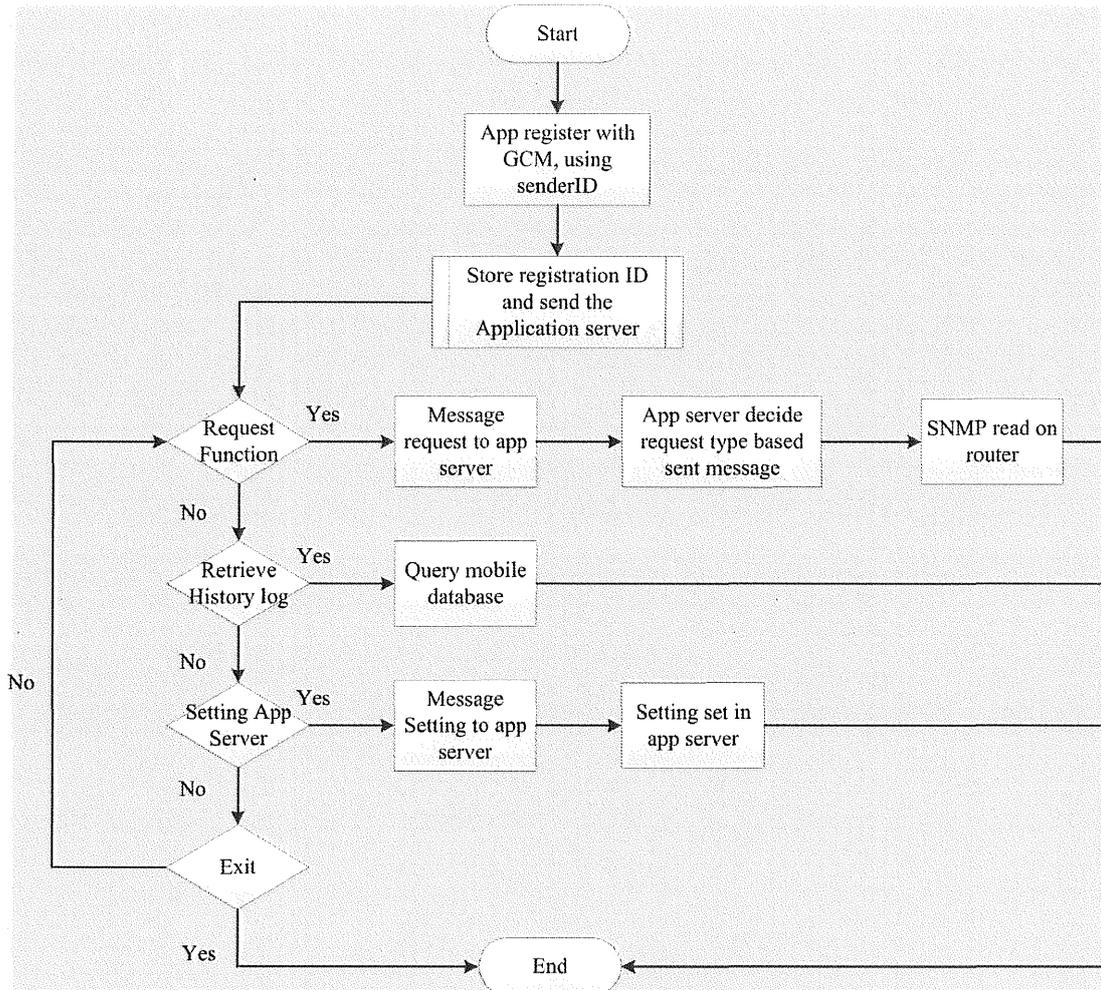


Figure 3.2: Overall System Flowchart

In the figure 3.2, show the flow where mobile application will start register with the Google Cloud Messaging. After successful obtain the ID which identify the mobile device, then user can use the function inside the application to interact with the application server which act as the intermediate between the mobile application and router. In the flow chart contain three main functions which are Request, Retrieve and Setting of the App Server. Request function is use to obtain the Router Status, Retrieve function is use to obtain the history log message and the Setting App Server function is use to set the status polling interval and exceed bandwidth usage notification.

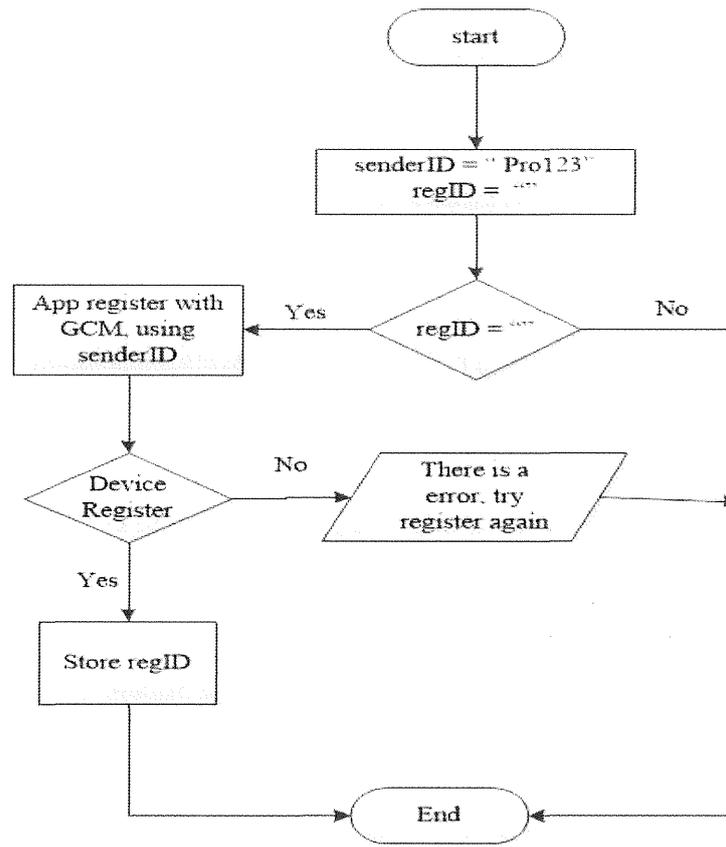


Figure 3.3: App Client register with GCM Server

In figure 3.3, is the flow chart on how the app client is register with the GCM server. In order to register with GCM server, app client need to have one variable to store the sender ID which is require to communicate with the GCM server. Besides, app client also need one variable to store the registration ID that GCM send back to app client and this registration ID is require to send to GCM server. Registration ID is required by app server to locate where to send the message.

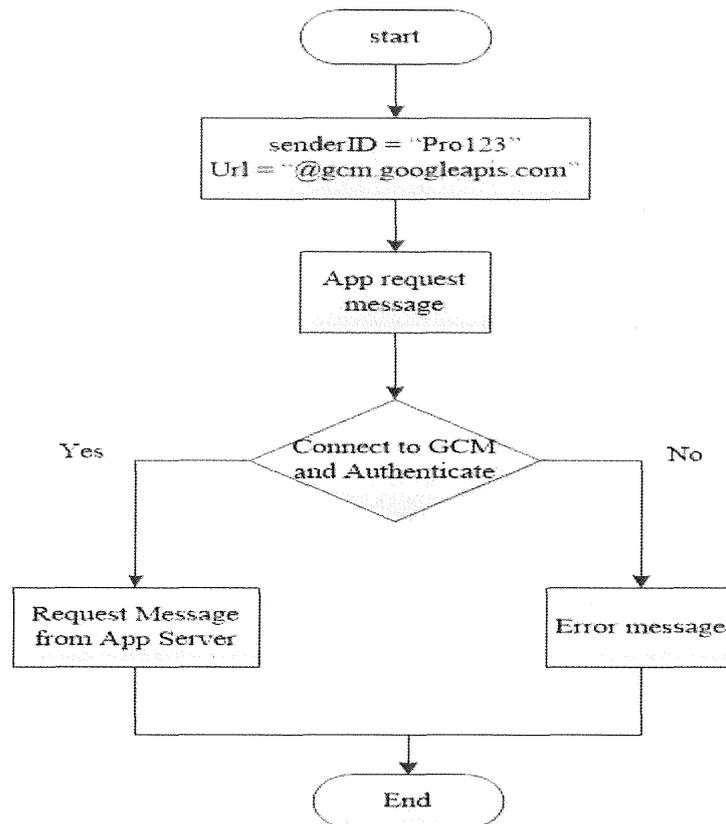


Figure 3.4: App client request message with App Server

In the figure 3.4, is the flow chart for app client to requests the message from app server through the GCM connection. In order to success create the GCM connection to request the message, app client need the address of the GCM server and the sender ID. The address is use as the destination for app client to post the request message and the sender ID is use as the authenticate credential to allow the GCM server send the correct message from app client to app server. If the connection from the app client and GCM server is success, then the request message from the app client will be send to the app server. If the connection is fail, then GCM will send the error message back to the app client.

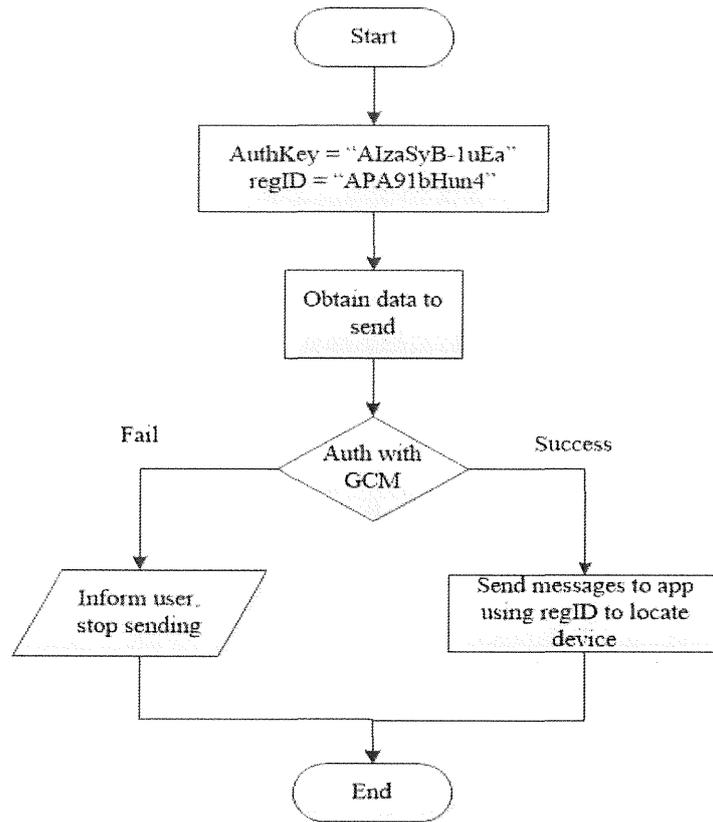


Figure 3.5: App Server send message to App Client

In the figure 3.5, is the flow chart for app server to send the message to the app client. To send the message, app server must have two variables that needed to put in the header of the message. The first one is the authentication key which is the API key of the GCM, this is to ensure that the authorize communication between the app server and GCM server. The app server will obtain the data from the database and then send to GCM, the GCM will use the authentication key in the message to ensure the integrity of the data before it send to the app client. If GCM fail to authorize the data, the message will be drop and the app server will be notify. If GCM success to authorize the data, the message will be send to app client using the registration ID to locate where the message should be send.

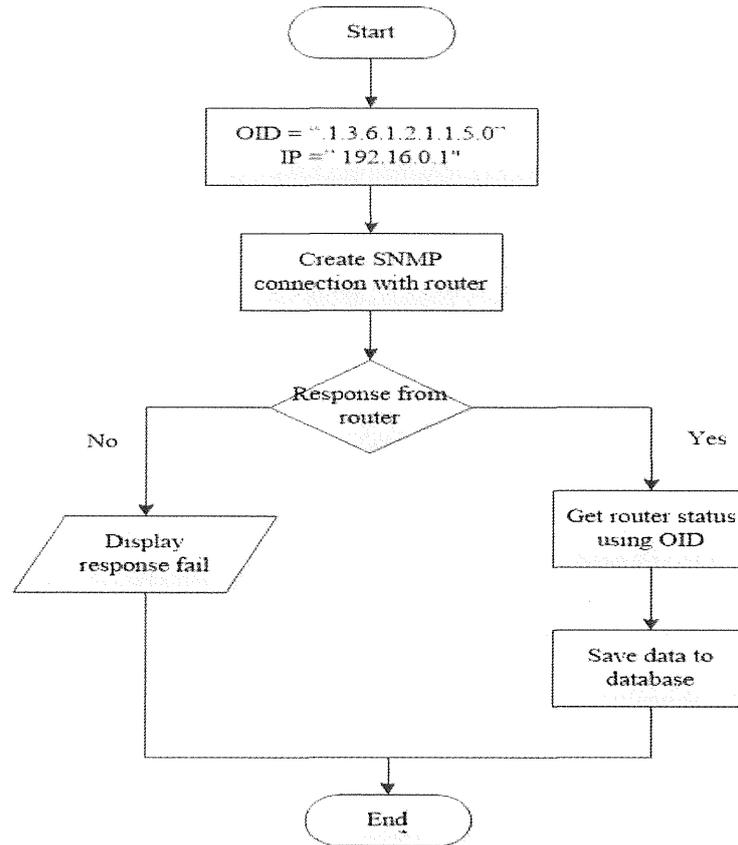


Figure 3.6: App Server gathers status information from router

In the figure 3.6, is the flow on how to gathers the status information from the router. To gathers the information from the router, must have the variables that store the Object Identified (OID) which is the numeric value that could use by SNMP to retrieve the information in the router. In order to gathers router information, app server must establish the SNMP connection with router. If the router is SNMP enable, router will be respond to the request and then app server can send the SNMP read command using the OID to gathers the router status information. When the information is obtain from the router, app server will save it to the database and after that send to the app client.

### 3.5 Database Design

This section describes the database design for router monitoring using android application. Below are the two database table 3.1 and 3.2 that will develop in the app client and app server.

Table 3.1: Database table in App client

| ATTRIBUTES | DEFINITIONS                                       | CONSTRAINTS    |
|------------|---|----------------|
| ID         | Auto Increment                                    | NUMBER         |
| CreateDate | Define date information of the router being saved | VARCHAR2 (100) |
| CreateTime | Define time information of the router being saved | VARCHAR2 (100) |
| Data       | Define status information of the router obtain    | VARCHAR2 (100) |

### 3.6 GUI Design

This section will describe the graphical user interface that will be implementing in the app client.



Figure 3.7: Main menu of the application

Figure 3.7 is the screen that will show up when open the app client, in this interface there will be three navigation button that link to other interface. The first button is the “Request” button, this button will bring user to the interface that contain different options for user to select the type of information user want to request. The second button “History”, this button will bring user to the interface that displays the history log message that being saved into the database. The third button “Setting”, this button will bring user to interface that allow the user to set the polling interval and exceed bandwidth utilization notification.

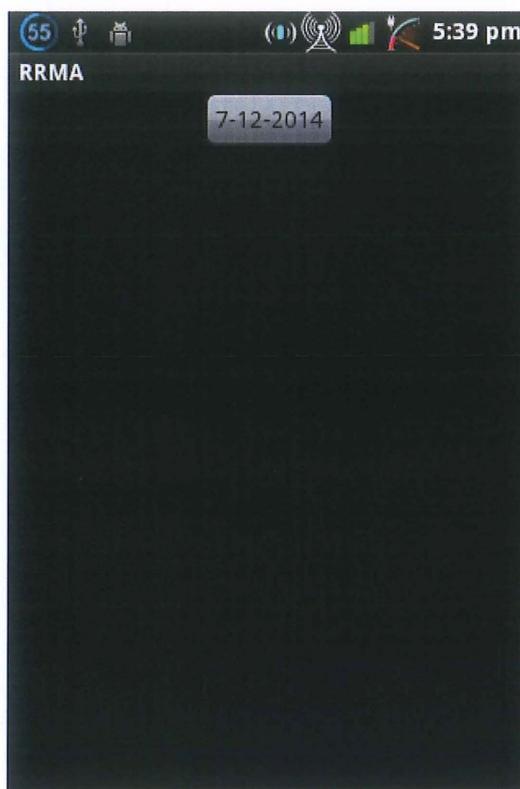


Figure 3.8: Interface to retrieve history log message

Figure 3.8 shows the interface that help user to retrieve the log message being receive from the app server. In this interface got one button that allows the user to select the date in which user want to view the saved log message. After press the button user will allow to select the require date after that the saved log message for that day will display in this interface. But when the date that user select didn't contain any data, then user will got message display to tell that no data for the date being select.



Figure 3.9: Interface that show basic information of the router

Figure 3.9 show the interface that displays the router Ethernet status information of the router east. This interface is where user can view the status information that retrieve from the router and being send to the app client. The information that sends to app client will be save inside the app client database, so when user close the application the information will not delete. Therefore user can always use this interface to check the basic information of the router that obtains from the app server.

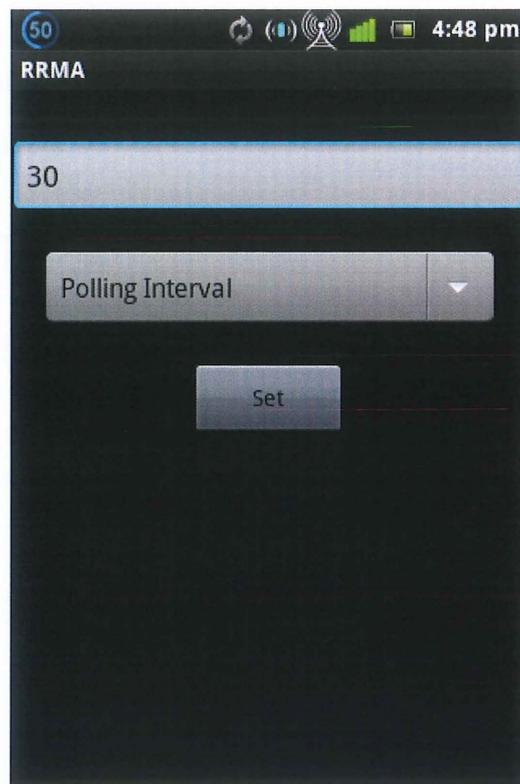


Figure 3.10: Interface for Setting the App Server

Figure 3.10 shows the interface that user can use to set the time interval where app server will automatically generate the router status and send to the mobile application. This interface also allows the user to set the notification in which the bandwidth utilization for certain port is reach. This interface is help to obtain the setting information and push to app server to implement.

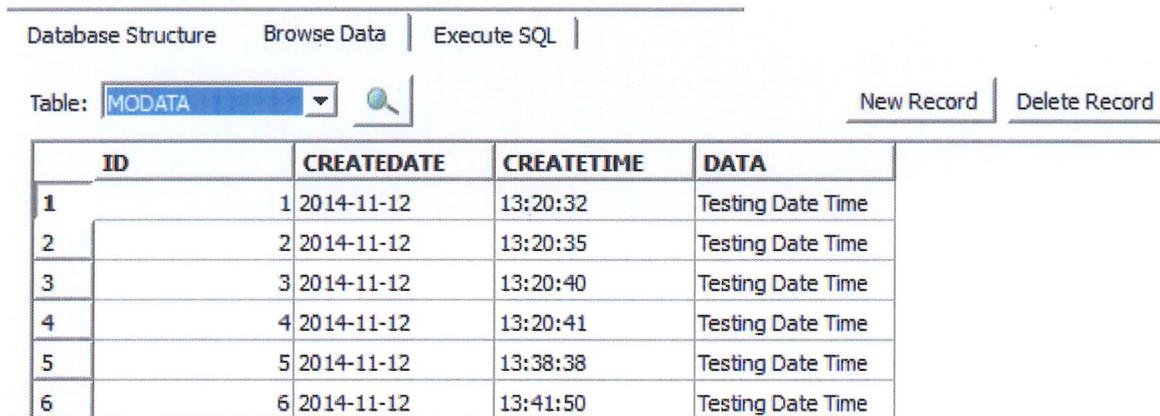
## CHAPTER 4

### IMPLEMENTATION

#### 4.1 Introduction

The purpose of this chapter is to discuss the process and coding that are implemented to develop the complete running application. In this section will include the important function and method that are using to create the database for storing of information and the steps to implement the server side and mobile application side. The complete process of database implementation is in section 4.2 and step and java code to develop the application server and mobile application are in section 4.3.

#### 4.2 Database Development



The screenshot shows a database management interface with the following elements:

- Navigation tabs: Database Structure, Browse Data, Execute SQL.
- Table selection: Table: MODATA (with a search icon).
- Actions: New Record, Delete Record.
- Table structure:

| ID | CREATEDATE   | CREATETIME | DATA              |
|----|--------------|------------|-------------------|
| 1  | 1 2014-11-12 | 13:20:32   | Testing Date Time |
| 2  | 2 2014-11-12 | 13:20:35   | Testing Date Time |
| 3  | 3 2014-11-12 | 13:20:40   | Testing Date Time |
| 4  | 4 2014-11-12 | 13:20:41   | Testing Date Time |
| 5  | 5 2014-11-12 | 13:38:38   | Testing Date Time |
| 6  | 6 2014-11-12 | 13:41:50   | Testing Date Time |

Figure 4.1: Mobile Application Database

The database name MODATA is created using the SQLiteDatabase as show in the Figure 4.1. In the database have four columns which are ID, CREATEDATE, CREATETIME and DATA. ID column is auto increment field which generated by the table, where CREATEDATE, CREATETIME and DATA column is use to save the summary information obtain from the router.

## 4.3 Application Development

### 4.3.1 Application Server

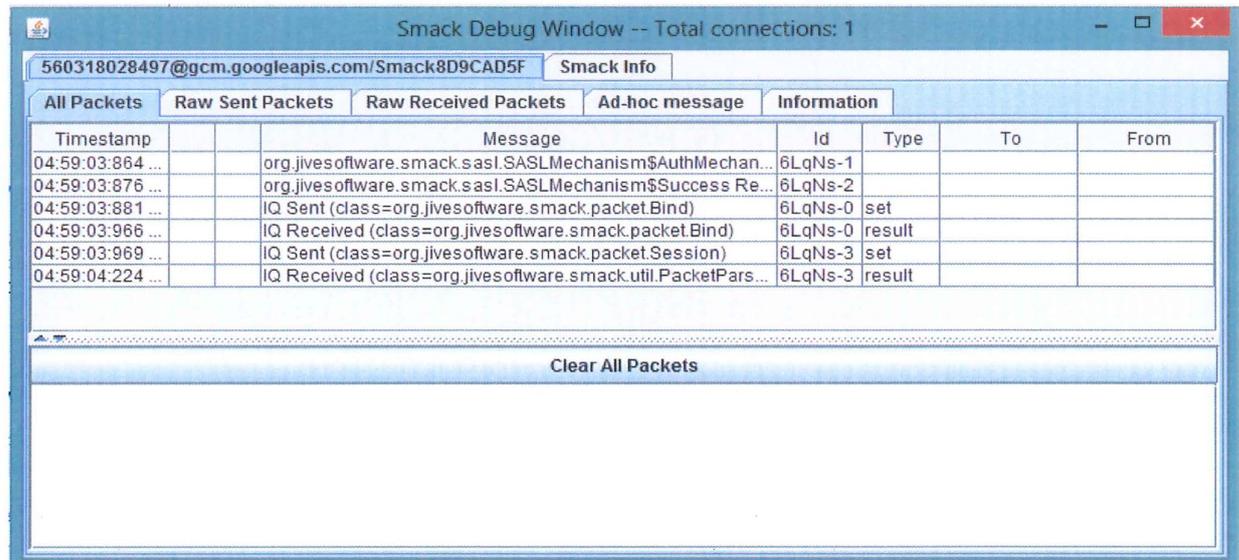


Figure 4.2: Application Server Debug Interface

To develop the application server, the project is using the smack API library to connect to the Google Cloud Messaging (GCM) Server. With the success of establish the connection, server and the phone will be able to communicate with each other. Figure 4.2 showing the smack debug window with information about packets sent and receive.

To connect to the GCM server, create one java class with method “void connect(long senderId, String apiKey)” as show in the Figure 4.3 where the senderID and apiKey would the credential that obtain from the google developer console when we create the project. In this method, using the XMPP properties that include in the smack API library then declaration of connection properties can be create by making this statement “connection = new XMPPTCPConnection(config);”. After that, using this declaration to start the connection “connection.connect();”.

```

public void connect(long senderId, String apiKey)
    throws XMPPException, IOException, SmackException {

    ConnectionConfiguration config = new ConnectionConfiguration(GCM_SERVER,
GCM_PORT);

    config.setSecurityMode(SecurityMode.enabled);

    config.setReconnectionAllowed(true);

    config.setRosterLoadedAtLogin(false);

    config.setSendPresence(false);

    config.setSocketFactory(SSLConnectionFactory.getDefault());

    config.setDebuggerEnabled(true);

    connection = new XMPPTCPConnection(config);

    connection.connect();

    connection.addConnectionListener(new LoggingConnectionListener());

    connection.addPacketListener(new PacketListener() {
        @Override
        public void processPacket(Packet packet) {
            Message incomingMessage = (Message) packet;

            GcmPacketExtension gcmPacket = (GcmPacketExtension) incomingMessage.
                getExtension(GCM_NAMESPACE);

            String json = gcmPacket.getJson();

```

Figure 4.3: GCM Connection Code

In order to handle the package that sent and receive, need to include this method “ public void processPacket(Packet packet){} “ as show in Figure 4.3 and in this method we can call another function to process the package depending on the message type we receive from the GCM server for example, acknowledge message, error message or standard upstream message.

```

try {
    @SuppressWarnings("unchecked")
    Map<String, Object> jsonObject = (Map<String, Object>) JSONValue.
        parseWithException(json);
    Object messageType = jsonObject.get("message_type");
    if (messageType == null) {
        handleUpstreamMessage(jsonObject);
        String messageId = (String) jsonObject.get("message_id");
        String from = (String) jsonObject.get("from");
        String ack = createJsonAck(from, messageId);
        send(ack);
    } else if ("ack".equals(messageType.toString())) {
        handleAckReceipt(jsonObject);
    } else if ("nack".equals(messageType.toString())) {
        handleNackReceipt(jsonObject);
    } else if ("control".equals(messageType.toString())) {
        handleControlMessage(jsonObject);
    } else {
        logger.log(Level.WARNING, "Unrecognized message type (%s)",
            messageType.toString());
    }
} catch (ParseException e) {
    logger.log(Level.SEVERE, "Error parsing JSON " + json, e);
} } }, new PacketTypeFilter(Message.class));
connection.login(senderId + "@gcm.googleapis.com", apiKey); }

```

Figure 4.4: GCM Connection Code

In Figure 4.4, method “Object messageType = jsonObject.get(“message\_type”)” will be call to extract the message type and if else function will use to compare the message type. Then the message will be send to the define function to handle the receive package.

```

protected void handleUpstreamMessage(Map<String, Object> jsonObject) {
    String category = (String) jsonObject.get("category");
    String from = (String) jsonObject.get("from");    this.setRedID(from);
    @SuppressWarnings("unchecked")
    Map<String, String> payload = (Map<String, String>) jsonObject.get("data");
    payload.put("ECHO", "Application: " + category);
    String echo = createJsonMessage(from, nextMessageId(), payload,
        "echo:CollapseKey", null, false);
    try {    sendDownstreamMessage(echo);
        } catch (NotConnectedException e) {
        logger.log(Level.WARNING, "Not connected anymore, echo message is not sent", e);
    } }

```

Figure 4.5: Handle Message Code

To handle the upstream message that send from the device, create “void handleUpstreamMessage(Map<String, Object> jsonObject){}” as show in the Figure 4.5 where jsonObject is the message that send from the phone and extract from the “void processPacket(Packet packet){}” that discuss early. Map<String, String> is use to declare the message payload format when create the data packet to be send to other device. In Figure 4.5, Json message will be extracted to be process and in the Json message have information about which phone application sending the data packet that stated in category and from which registration ID as different phone have different registration ID get from the GSM server.

In Figure 4.6, “sendDownstreamMessage(String jsonRequest)” function is use to send an message back to device to inform the success of sending message from the device and this function will be call by “sendDownstreamMessage(echo);”

In Figure 4.7, “createJsonMessage()” function is use to combine all the necessary information in one packet before sending to GCM server. This function will be call by “createJsonMessage(from, nextMessageId(), payload, "echo:CollapseKey", null, false);

```

public boolean sendDownstreamMessage(String jsonRequest) throws
    NotConnectedException {
    if (!connectionDraining) {
        send(jsonRequest);
        return true;    }
    logger.info("Dropping downstream message since the connection is draining");
    return false;
}

protected void send(String jsonRequest) throws NotConnectedException {
    Packet request = new GcmPacketExtension(jsonRequest).toPacket();
    connection.sendPacket(request); }

```

Figure 4.6: Send message Code

```

public static String createJsonMessage(String to, String messageId, Map<String, String>
    payload, String collapseKey, Long timeToLive, Boolean
    delayWhileIdle) {
    Map<String, Object> message = new HashMap<String, Object>();
    message.put("to", to);
    if (collapseKey != null) {
        message.put("collapse_key", collapseKey);    }
    if (timeToLive != null) {
        message.put("time_to_live", timeToLive);    }
    if (delayWhileIdle != null && delayWhileIdle) {
        message.put("delay_while_idle", true);    }
    message.put("message_id", messageId);
    message.put("data", payload);
    return JSONValue.toJSONString(message); }

```

Figure 4.7: Create Data Message Code

### 4.3.2 Mobile Application

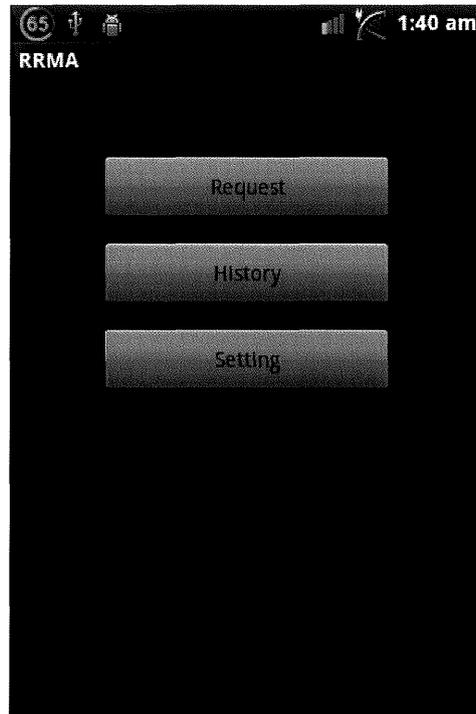


Figure 4.8: Main Menu Interface

Figure 4.8 show the main interface of the mobile application that contain “Request”, “History” and “Setting” button. “Request” button is use to send the request message to server and display the message that send from the server. In order to create the communication between the server and mobile application, mobile application needs to register with the GCM server and get the registration ID and send to the server.

To register and get the registration ID, need to create function “void registerInBackground() {}” as show in the Figure 4.9. This function will call “gcm.register(SENDER\_ID);” to register this device in the GCM server by using the sender ID that obtain from the google develop console.

```

private void registerInBackground() {
    new AsyncTask<Void, Void, String>() {
        @Override
        protected String doInBackground(Void... params) {
            String msg = "";
            try {
                if (gcm == null) {
                    gcm = GoogleCloudMessaging.getInstance(context);
                }
                regid = gcm.register(SENDER_ID);
                msg = "Device registered, registration ID=" + regid;
                Log.i(TAG, "Register ID : " + regid);

                sendRegistrationIdToBackend();

                storeRegistrationId(context, regid);
            } catch (IOException ex) {
                msg = "Error :" + ex.getMessage();
            }
            return msg;
        }

        @Override
        protected void onPostExecute(String msg) {
            mDisplay.append("\n " + msg);
        }
    }.execute(null, null, null);
}

```

Figure 4.9: GCM registration code

When the device get the registration ID from the GCM server, method “storeRegistrationId(context, regid);” as show in Figure 4.9 will be used to store the registration ID obtain so the device will not need to register again in the future.

Again the registration ID need to send to the application server in order to allow the application server to recognize which device it has connection to. This process will be done by calling the function “sendRegistrationIdToBackend();” as show in the Figure 4.10 . Then one message containing the registration ID will be send to application server.

```

private void sendRegistrationIdToBackend() {
    Log.d(TAG, "REGISTER USERID: " + regid);
    String name = "Dong";
    new AsyncTask<String, Void, String>()
    {
        @Override
        protected String doInBackground(String... params)
        {
            String msg = "";
            try
            {
                Bundle data = new Bundle();
                data.putString("name", params[0]);
                data.putString("action", "1234testing");
                String id = Integer.toString(msgId.incrementAndGet());
                Long timeToLive = 10000L;
                gcm.send(SENDER_ID + "@gcm.googleapis.com", id, timeToLive,
data);

                msg = "Sent registration";
            }
            catch (IOException ex)
            {
                msg = "Error ." + ex.getMessage();
            }
            return msg;
        }

        @Override
        protected void onPostExecute(String msg)
        {
            Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
        }
    }.execute(name);
}

```

Figure 4.10: Send Registration ID to App Server



Figure 4.11: Request Router Information Interface

Figure 4.11 shows the request interface of the mobile application when user clicks on the “Request” button as show in Figure 4.8. The text box in this interface is use to insert the IP address to identify which router user need to request the information. When user finish insert the IP address, “Request” button is use to send the request message to server and display the message that send from the server. In next page will describe the detail process in the back end coding that performs by this button.

```

public void onClick(final View view) {

    if (view == findViewById(R.id.send)) {
        new AsyncTask<Void, Void, String>() {
            @Override
            protected String doInBackground(Void... params) {
                String msg = "";
                try {
                    Bundle data = new Bundle();
                    data.putString("my_message", "Hello World 1234");
                    data.putString("my_action",
                        "com.google.android.gcm.demo.app.ECHO_NOW");
                    String id = Integer.toString(msgId.incrementAndGet());
                    gcm.send(SENDER_ID + "@gcm.googleapis.com", id, data);
                    msg = "Sent message";
                } catch (IOException ex) {
                    msg = "Error :" + ex.getMessage();
                }
                return msg;
            }

            @Override
            protected void onPostExecute(String msg) {
                mDisplay.append("\n " + msg);
            }
        }.execute(null, null, null);
    } else if (view == findViewById(R.id.clear)) {
        mDisplay.setText("");
    }
}

```

Figure 4.12: Handle Button Click Event

When the user click on the “Request” button, “onclick” function as show in the Figure 4.12 will create the data packet and send to the server. Method “Bundle data = new Bundle();” is use to create one data packet and “data.putString” is use to input the message that want to send into the data packet. When the data packet is ready to be send, it will call “gcm.send” to send the packet to server.

```

protected void onHandleIntent(Intent intent) {

    Bundle extras = intent.getExtras();
    GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(this);
    String messageType = gcm.getMessageType(intent);

    if (!extras.isEmpty())
    (GoogleCloudMessaging.MESSAGE_TYPE_SEND_ERROR.equals(messageType)) {
        sendNotification("Send error: " + extras.toString());
    } else if
    (GoogleCloudMessaging.MESSAGE_TYPE_DELETED.equals(messageType)) {
        sendNotification("Deleted messages on server: " + extras.toString());
    } else if
    (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equals(messageType)) {
        Log.i(TAG, "Completed work @ " + SystemClock.elapsedRealtime());

        String message = intent.getExtras().getString("message");

        if (message == null) {
        }
        else{
            sendNotification("Received: " + message);
        }
        Log.i(TAG, "Received: " + extras.toString());
    }
}
GcmBroadcastReceiver.completeWakefulIntent(intent);
}

```

Figure 4.13: Handle Intent Event

When phone receive the message from the GCM server, intent will be create in order to process the message and put into the notification. Function “onHandleIntent” as show in the Figure 4.13 will get the message type of GCM message being send and this function will compare to the GCM message type that have been declare in the API. If the message type is just the common message, it will extract the message and call “sendNotification” function as show in the Figure 4.14 to create notification in the phone.

```

private void sendNotification(String msg) {
    mNotificationManager = (NotificationManager)
        this.getSystemService(Context.NOTIFICATION_SERVICE);

    Intent intent = new Intent(this, DemoActivity.class);
    intent.putExtra(DemoActivity.EXTRA_MESSAGE, msg);
    PendingIntent contentIntent = PendingIntent.getActivity(this, 0, intent, 0);

    NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_stat_gcm)
            .setContentTitle("GCM Notification")
            .setStyle(new NotificationCompat.BigTextStyle().bigText(msg))
            .setContentText(msg)
            .setTicker(msg)
            .setAutoCancel(true)

        .setSound(Settings.System.DEFAULT_NOTIFICATION_URI);

    mBuilder.setContentIntent(contentIntent);
    mNotificationManager.notify(NOTIFICATION_ID, mBuilder.build());
}

```

Figure 4.14: Create Notification Message

Notification that pushes to phone is create using the function “NotificationCompat.Builder” as show in the Figure 4.14, inside this function the property of the notification is declares by using method “.set”. “.setContentText” will display the content of the message to the user where “.setContentTitle” will set the notification title when the notification is push to the user.

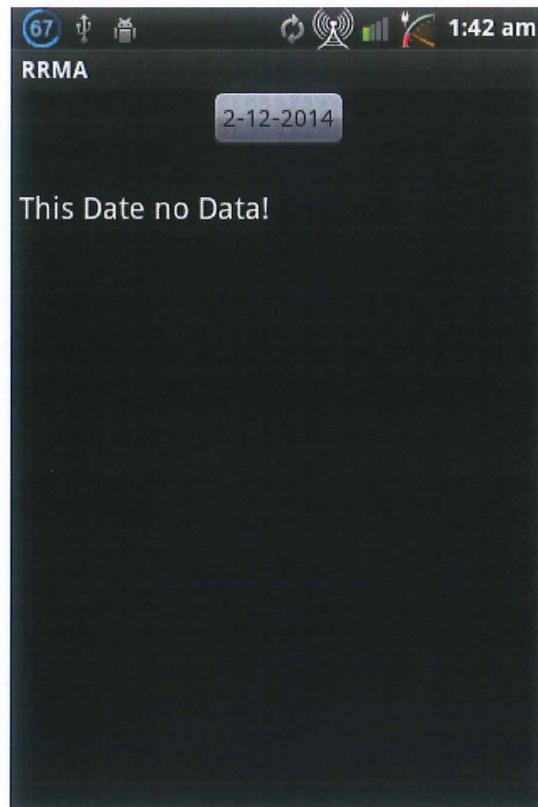


Figure 4.15: Retrieve History Log message Interface

Figure 4.15 shows the request interface of the mobile application when user clicks on the “History” button as show in Figure 4.8. The time selection box in this interface is use by user to set which date to display the data that have been save in the database. In next page will describe the detail process in the back end coding that performs by this selection button.

```

final DatePickerDialog.OnDateSetListener datePickerListener = new
DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year, int month, int day) {

        assignmentdue.set(Calendar.YEAR, year);
        assignmentdue.set(Calendar.MONTH, month);
        assignmentdue.set(Calendar.DAY_OF_MONTH, day);

        btnRetrie.setText(Integer.toString(day)+"-"+Integer.toString(month+1)+"-
"+Integer.toString(year));

        String str = Integer.toString(year)+"-"+ Integer.toString(month+1)+"-"+ Integer.toString(day);

        ArrayList<String> contactList = new ArrayList<String>();
        ArrayList<String> contactList1 = new ArrayList<String>();
        mDisplay.setText("");

        contactList = ramsDataBaseAdapter.getEntry(str);
        contactList1 = ramsDataBaseAdapter.getData(str);
        int i = contactList.size();

        if(i == 0){
            mDisplay.setText("\n This Date no Data!");
        }

        for(int j=0;j<i;j++){
            mDisplay.append("\n " + contactList.get(j) + "," + "\n  " + contactList1.get(j) + ";\n");
        }

        }};

```

Figure 4.16: Handle Date Selection

Figure 4.16 is backend coding for the selection button in interface Figure 4.15. DatePickerDialog is use to create a pop up screen to let user select the date, and assignmentdue.set will use to set the date being select to show on the button. ramsDataBaseAdapter.getData is the function use to extract the specific data on the specific date that has been select by the user.

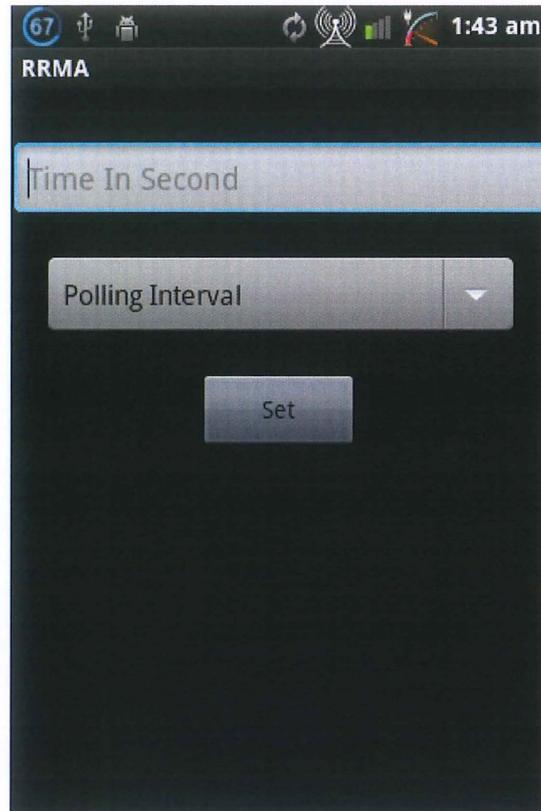


Figure 4.17: Setting App Server Interface

Figure 4.17 shows the setting interface of the mobile application when user clicks on the “Setting” button as show in Figure 4.8. The text box in this interface is use to insert the time interval to be set in the app server. When user finish insert the time in seconds, “Set” button is use to send the time setting to server and then the app server will configure with the time interval being accept as the interval for auto generate the router status information. In next page will describe the detail process in the back end coding that performs by this button.

```

public void onClick(final View view) {

    if (view == findViewById(R.id.btnsetTI)) {
        new AsyncTask<Void, Void, String>() {

            String timeorper = String.valueOf(etextTI.getText());
            @Override
            protected String doInBackground(Void... params) {
                String msg = "";
                try {
                    Bundle data = new Bundle();
                    data.putString("dataTo", timeorper);
                    data.putString("my_request", reqType);
                    data.putString("from_page", "settingPage");
                    data.putString("my_action", "com.android.gcm.rams.app.ECHO_NOW");

                    String id = Integer.toString(msgId.incrementAndGet());
                    gcm.send(SENDER_ID + "@gcm.googleapis.com", id, data);
                    msg = "Sent message";
                } catch (IOException ex) {
                    msg = "Error ." + ex.getMessage();
                }
                return msg;    }
            @Override
            protected void onPostExecute(String msg) {
                mDisplay.append("\n " + msg);
            } }
    }
}

```

Figure 4.18: Handle Setting Button

Figure 4.18 is backend coding for the “Set” button in interface Figure 4.17. If() function is use to detect which button has been click using ID “view == findViewById(R.id.btnsetTI)”. “Bundle data = **new** Bundle()” is use to create the data to be send to the app server, where variable timeorper in “data.putString(“dataTo”, timeorper)” contain the time interval that capture from the text box in figure 4.17 which will then be configure in the app server.

#### 4.4 Testing Application

In this section will showing the data that obtain after running the application.



Figure 4.19: Receive Notification

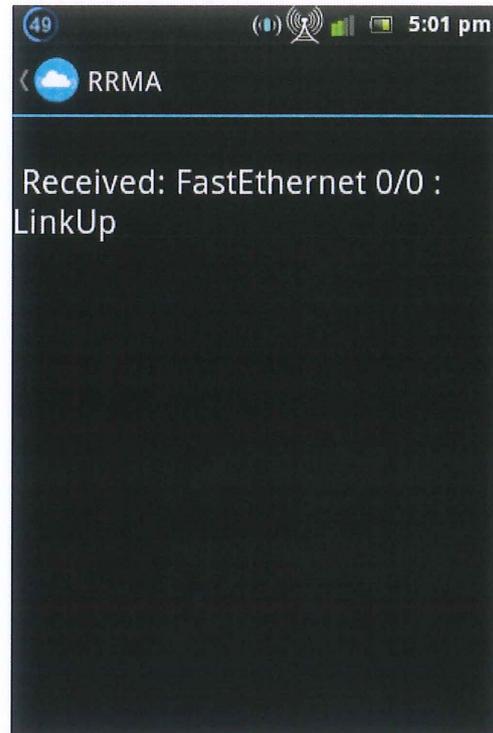


Figure 4.20: Display log Message

After user request the router information using the interface that show in Figure 11, the application server will extract the information from router and then send to the mobile application. When user receives the data message, notification such as Figure 4.19 will be show in our notification bar. User can click on the notification to open up the data message that being sent by the application server, then one interface such as Figure 4.20 will be showing up and display the data message that being received.

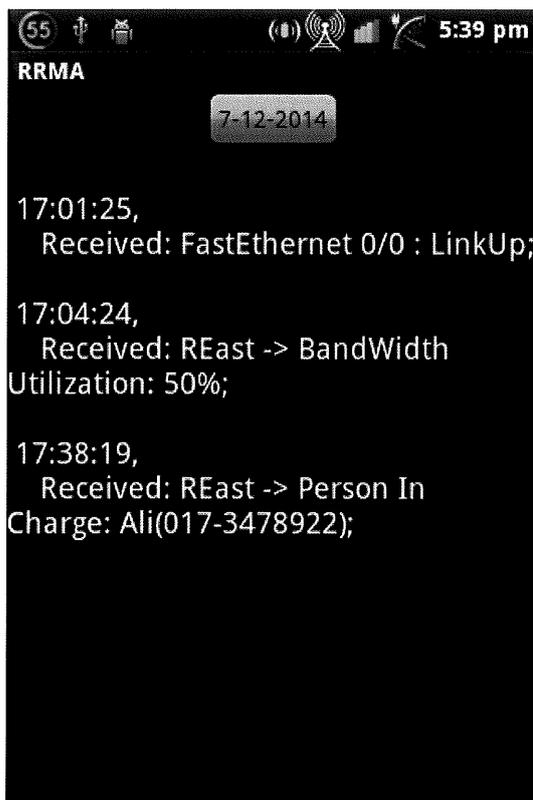
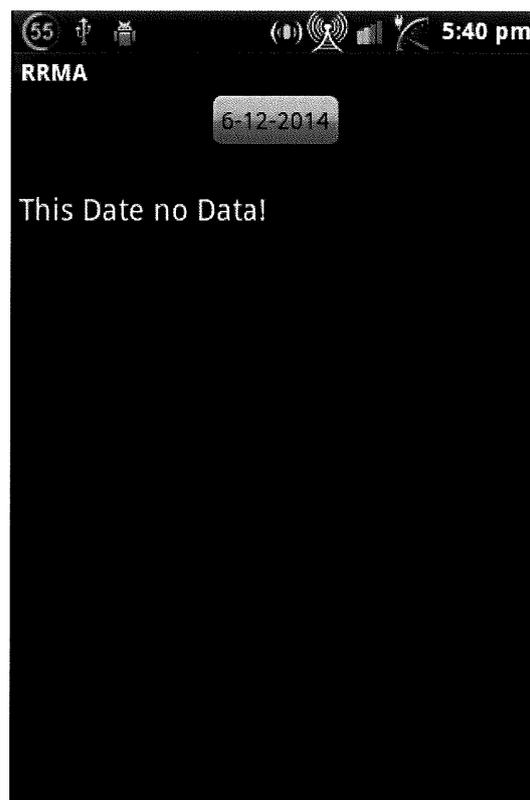


Figure 4.21: Display History Log Message

Figure 4.22: Date with no Log  
Message Saved

This application contains the database to save the log message that being received previously. When user click on the date selection button, the log message that being save in the database will be retrieve and display such as Figure 4.21. But when the date that being select don't have any log message saved, then user will be informing such as Figure 4.22.



Figure 4.23: Reboot Request

Figure 4.24: Reboot Success  
Notification

By using this mobile application, user also can remotely reboot the router but in prerequisite of enter the correct security code for security verification. In order to remotely reboot the router, must have correct IP address and security code enter in Figure 4.23. Then the application server will verify the input credentials, if correct then router will reboot and success reboot message will be send to inform the user such as Figure 4.24.



Figure 4.25: Request System up Time

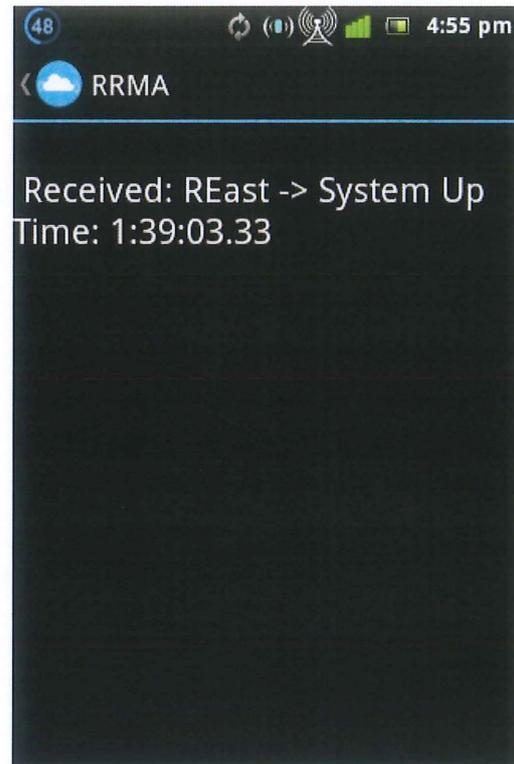


Figure 4.26: Display System up Time

After user request the router up time information using the interface that show in Figure 4.25, the application server will extract the information from router and then send to the mobile application. When user receives the data message, notification will show up in the notification bar. User can click on the notification to open up the data message that being sent by the application server, then one interface such as Figure 4.26 will be showing up and display the router system up time.

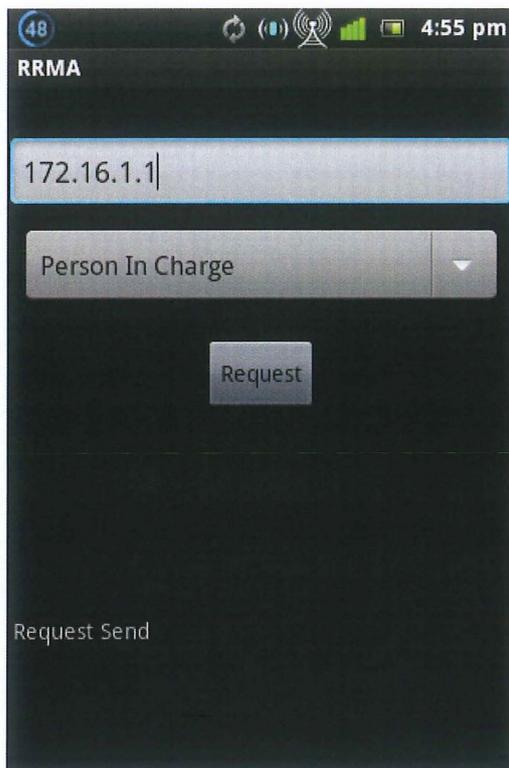


Figure 4.27: Request Person in Charge

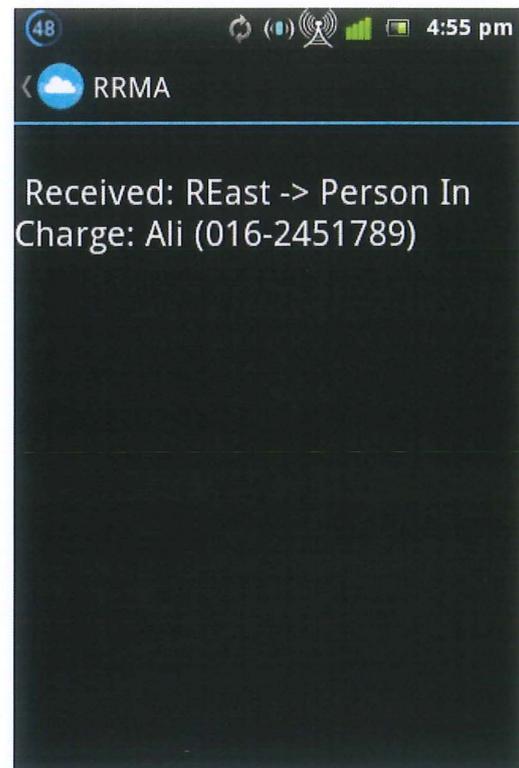


Figure 4.28: Display Person in Charge

After user request the Person in Charge information using the interface that show in Figure 4.27, the application server will extract the information from router and then send to the mobile application. When user receives the data message, notification will show up in the notification bar. User can click on the notification to open up the data message that being sent by the application server, then one interface such as Figure 4.28 will be showing up and display the Person in Charge for the router.

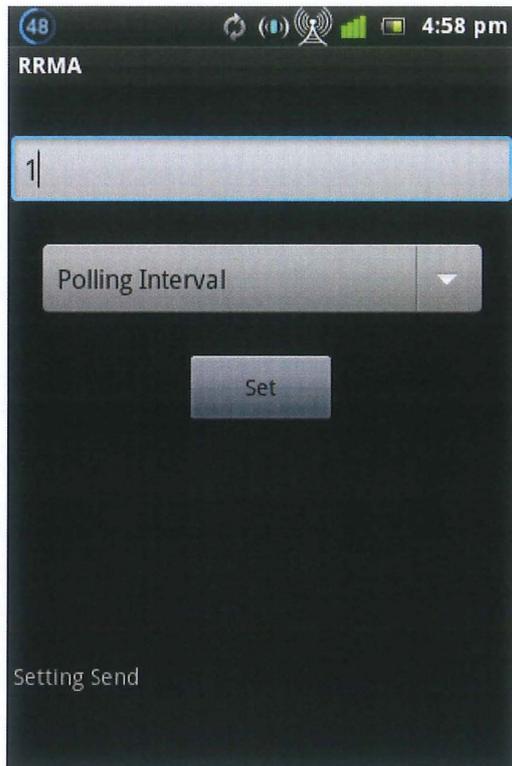


Figure 4.29: Setting Polling Interval

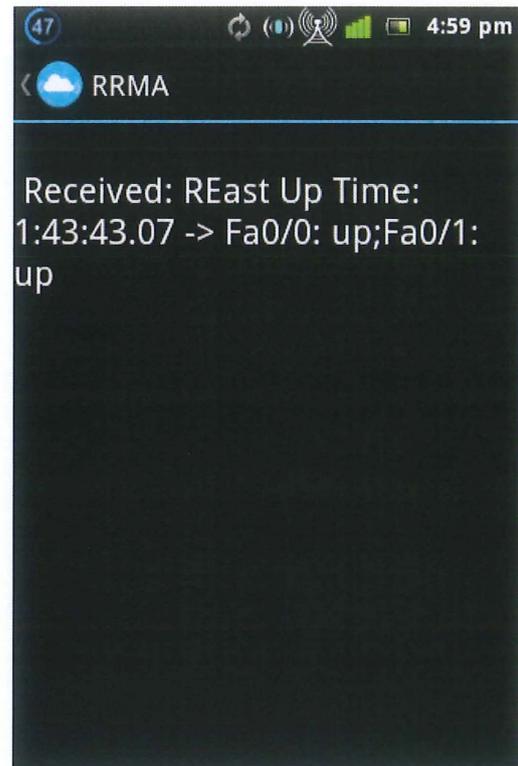


Figure 4.30: Message receive after 1 minute

After user set the polling interval to automatically generate the router status information using the interface that show in Figure 4.29, the application server will be declare the obtain interval to pull the router status information and send to the mobile application. When user receives the data message, notification will show up in the notification bar. User can click on the notification to open up the data message that being sent by the application server, then one interface such as Figure 4.30 will be showing up and display all the router information.

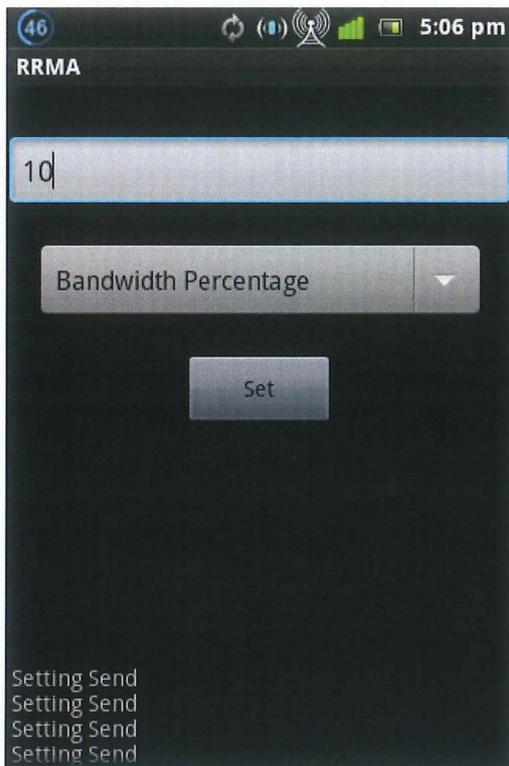


Figure 4.31: Setting Bandwidth Usage Notification

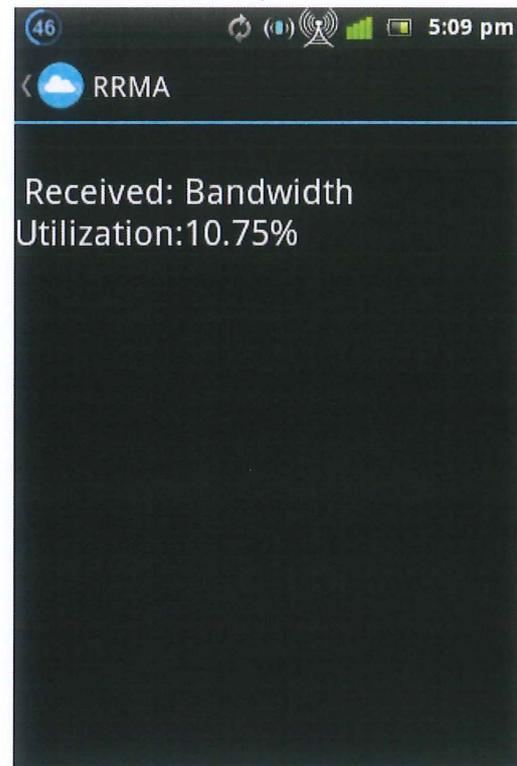


Figure 4.32: Exceed Bandwidth Usage Notification

After user set the exceed bandwidth notification to automatically alert the user when bandwidth utilization is reach using the interface that show in Figure 4.31, the application server will be declare the bandwidth utilization limit then app server will automatic calculate the bandwidth utilization for certain port. When the bandwidth utilization limit set is reach, the app server will send the alert message to the mobile application. When user receives the data message, notification will show up in the notification bar. User can click on the notification to open up the data message that being sent by the application server, then one interface such as Figure 4.32 will be showing up and display exceeded bandwidth utilization.

## CHAPTER 5

### RESULT AND DISCUSSION

#### 5.1 Introduction

In this chapter will discuss about the result after develop and testing of the Router Remote Monitoring Using Android Application, the advantages and disadvantages, the constraints of this application.

After this application was testing, the developer found out that this application meets the objective that stated in chapter 1 before which is:

- i. To develop an application on the palm of the hand that cans remotely viewing the status of router.
- ii. Develop an application server to help network administrator to automatic generate the report of router status.
- iii. Alert the user about the network problem that occurs.

## **5.2 Advantages and Disadvantages**

Any application development must have its own advantages and disadvantages. On this application, the developer will explain briefly about the application advantages and disadvantages.

### **5.2.1 Advantages**

Router Remote Monitoring Using Android Application developed was given advantages to the network administrator. Among the advantages are:

- i. The application can generated real time alert notification and inform network administrator when router has connection problem.
- ii. Network administrator can remotely reboot the router when the router bandwidth utilization is very high.

### **5.2.2 Disadvantages**

Although this application is achieves the objective, but there are still has the advantages and limitation. This application will not function as intended when the connection of the application server to the router is down which will cause the application server will not been able to generate the status information from the router and send to the mobile application. And this application has its limitation of data presentation, the mobile application which able to present the data in a message format but not the graphical view.

### **5.3 Constraints**

During the development process, the developer faces with few constraints. Among of the constraints are:

i. Limited of knowledge

The main constraint is about the knowledge on how to develop this application. This is because this application required the knowledge of java language and xml language which is not familiar by the developer. Plus the development of this application using a lot of java library and function which are new to the developer added the challenge in developing this application.

ii. Time constraint

To develop the application server, require the present of the router for testing. So developer only can use the lab free time to develop the application. Therefore, developer always needs to rush in the given short time to complete its application function and testing.

### **5.4 Summary**

The result of the Router Remote Monitoring Using Android Application are presented and discussed in this chapter. The following chapter will focuses on the conclusion and the suggestion for future enhancement.

## CHAPTER 6

### CONCLUSION

#### 6.1 Conclusion

This application “Router Remote Monitoring Using Android Application” is develop for network administrator to remotely getting the status information from the router send by the app server. The operation of the router is critical because failure of one networking device will eventually affect other network device operation.

This application provides the network administrator to keep track all the router information status from the palm of the hand. With this application, network administrator can react to the failure as soon as possible where the alert information of the router is noticeable in the phone. With this application, monitoring the router status information will become easy when network administrator is away from the workstation. An advantage of this application is that network administrator will not miss any networks failures that occur at any time.

There is still need improvement in order to increase the functionality of this application such as adding the performance report of the router. Besides, this application require one app server to direct the status information gathers from router to app client and this increase the complexity to implement the application. But this application is still helping the network administrator to keep track status of the router when they are away from the workstation, regardless of the limitation of functionality in this application. Therefore, this is application is still has its own strength as an application to report the router status to the network administrator.

## 6.2 Future Enhancement

Although this application already fulfill the objective, scope and purpose successfully, but it still have some limitations. Some of them are:

- i. The functionality of graph view with time polling interval.
- ii. The database of this application still needs to be enhanced.

This is the suggestions for the future application improvement:

- i. Add interface to show the data in graphical view where the time polling interval of the data also can be directly set at this interface.

## References

Ball, M., Magnanti, T., Monma, C., & Nemhauser, G. (1995). Network routing. Elsevier New York.

Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. Arxiv Preprint Arxiv:1205.6904.

Davin, J., Case, J., Fedor, M., & Schoffstall, M. (1990). Simple network management protocol (SNMP).

Davis, A., Bersoff, E., & Comer, E. (1988). A strategy for comparing alternative software development life cycle models. Software Engineering, IEEE Transactions On, 14(10), 1453--1461.

Gavalas, D. (2001). Mobile software agents for network monitoring and performance management.

Hansen, J., Gronli, T., & Ghinea, G. (2012). Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges. International Journal Of Communications, Network & System Sciences, 5(12).

Hansen, J., Gronli, T., & Ghinea, G. (2012). Cloud to device push messaging on Android: a case study, 1298--1303.

Harrington, D., Wijnen, B., Presuhn, R., & y,. (2002). An architecture for describing simple network management protocol (SNMP) management frameworks.

Jianqun, L. (2000). Applications of RAS Router base on PC in Remote Monitoring. Guangdong, 4, 009.

Lacy, L. (2012). *Software Development Process* (1st ed.). New Delhi: World Technologies.

Medhi, D., & Ramasamy, K. (2007). *Network routing* (1st ed.). Amsterdam: Elsevier/Morgan Kaufmann Publishers.

Miller, M. (1993). *Managing internetworks with SNMP* (1st ed.). New York, N.Y.: M & T Books.

Mizell, C., & Malone, L. (2009). A software development simulation model of a spiral process. Software Engineering Competence Center (SECC).

Nimodia, C., & Deshmukh, H. (2012). *Android Operating System*. Software Engineering, ISSN, 2229--4007.

Pohja, M. (2009). Server push with instant messaging, 653--658.

Reuter, E., & Baude, F. (2002). A mobile-agent and SNMP based management platform built with the Java ProActive library, 140--145.

Rogers, R. (2009). *Android application development* (1st ed.). Sebastopol, Calif.: O'Reilly.

Simao, E. (2009). *Software Development Methodologies*.

Waldbusser, S. (2006). *Remote network monitoring management information base version 2*.

Yanuar, K. (2012). *Social Networking Application integrates with Cloud Messaging using Google App Engine on Android*.

# Appendix

## Gantt Chart

