Design and Implementation of a Mobile Network Management Tool for Wireless Site Surveying

Kilan Muhammed Hussein

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
June 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr.Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Doğu Arifler
Supervisor

Examining Committee

1. Prof. Dr. Işık Aybay
2. Assoc. Prof. Dr. Doğu Arifler
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

"Everywhere Wi-Fi" across the campus of Eastern Mediterranean University (EMU) is a very positive and exciting experience for faculty, staff, students, and visitors. Having instant access to web sites, email, the Internet, and other IT services regardless of location can dramatically change the way of study and learning.

The increasing utilization of networks, especially wireless local area networks (WLANs), for different applications and in aspects of modern life, has resulted in a lot of studies towards the analysis and optimal design of WLANs. Deploying a large-scale, campus-wide Wi-Fi system presents many challenges that affect delivery of internet services. These challenges are managing, monitoring, coverage, capacity and density evaluation and security assessment.

In this study, we focus on the managing and monitoring aspects of the WLANs by designing and developing a software framework "EMUWiFiManager" as a mobile network management application on the Android platform. The application stores and displays basic information about the access points by using Simple Network Management Protocol (SNMP) and displays their locations on Google maps.

With the developed application, we conducted a measurement study and collected data for reporting and visualizing the network utilization of a campus wireless network.

Keywords: Wi-Fi access points, mobile networks, Android OS, network management

ÖZ

Doğu Akdeniz Üniversitesi'nde kampüs genelinde her yerde Wi-Fi bağlantısı olması, personel, öğrenciler ve ziyaretçiler için çok olumlu bir deneyimdir. Her yerde ve anında web sitelerine, e-posta, internet ve diğer bilgi teknoloji hizmetlerine erişim, çalışmanın ve öğrenmenin biçimini dramatik olarak değiştirebilmektedir.

Ağların, özellikle WLANların, artan kullanımı, farklı uygulamalar ve modern yaşamın özellikleri, WLANların analiz ve optimal tasarımına yönelik çalışmaları arttırmıştır. Büyük ölçekli yerleşke genelinde Wi-Fi sistemi dağıtma ve internet hizmetleri sunma birtakım sorunları ortaya çıkarmaktadır. Bu sorunlar arasında, ağ gözlemi, kapsama alanı, kapasite, yoğunluk değerlendirmesi ve güvenlik yönetimi vardır .

Bu çalışmada WLAN erişim noktalarının yönetim ve gözlemlemesi için "EMUWiFiManager" adında Android platformunda bir ağ yönetim yazılımı tasarlanıp geliştirilmiştir. Uygulama, erişim noktalarıyla ilgili temel bilgileri SNMP kullanarak sorgulayıp saklamakta ve coğrafi yerlerini ise Google Maps'ta göstermektedir.

Ayrıca, geliştirilen yazılımla bir ölçüm çalışması yapılmış ve örnek kampüs kablosuz ağ kullanımı raporlanıp ve görselleştirilmiştir.

Anahtar Kelimeler: Wi-Fi erişim noktaları, mobil ağlar, Android işletim sistemi, ağ yönetimi

To My Loving Mother

To My Loving wife

To My Loving brothers and sisters

To my friends who encouraged and supported me

ACKNOWLEDGMENT

First of all I would like to thank to our most beloved ALLAH, for supporting me in my master study and in this research particularly.

I express my deepest gratitude and respect to my supervisor Assoc. Prof. Dr. Doğu Arifler for his guidance and support throughout my research.

I am also thankful to all my teachers at EMU for their encouragement and to my colleagues who helped me during my studies.

I am also thankful to Diyala University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Program Interface
APS	Access Points
AVD	Android Virtual Machine
CDT	C/C++ Development Tooling
GSM	Global System for Mobile
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
J2SE	Java 2 Standard Edition
JDT	Java Development Tools
LIB	Library
MIB	Management Information Base
NMS	Network Management System
OHA	Open Handset Alliance
OID	Object Identifier
OTA	Over The Air
PDT	PHP Development Tools
RFC	Request for Comments
SDK	Software Development Kit
SNMP	Simple Network Management Protocol
SSID	Service set identification
WLAN	Wireless Local Area Network
XML	Extendable Markup Language

Chapter 1

INTRODUCTION

Wireless Local Area Networks (WLANs) are the best choice to build networks and with the availability of network solutions, WLANs have grown and this trend is expected to continue in the next years because they are easy to install and allow mobility for users without concerns of being disconnected. The implementations of 802.11n are quickly nearing to the speed of wired networks. Therefore, increased bandwidth has led to the wide adaptation of WLANs in the campus wireless network [1]. In contrast, increasing utilization of networks, especially WLANs, for different applications and in aspects of modern life, has led to more studies and attention towards the analysis and optimal design of WLANs [2].

Distinguishing features of the wireless environment and the distributed nature of the network setup have raised many important challenges in finding the performance limits of different tasks such as communication, control, and computations over WLANs. Also, many design methods concerning the complexity and the robustness of wireless systems should be addressed for a thorough understanding and an efficient operation of wireless networked systems [2]. Moreover, wireless applications of the new technologies have led to a series of new problems of wireless network communications so that monitoring of the wireless network effectively has become significant [3].

This chapter is devoted to the background of project , literature review and lastly, the aim of this work.

1.1 Backround

The advancement of technology has made life easier. The spread of mobile technologies enables us to constantly be in touch with the world. By it, more aspects of our lives are brought together for easy access. For example, a person could be making finishing touches with his presentation for next day, call phone, email, chat, check weather, check portal, book a flight, to mention a few, all in the same place and likely simultaneously [4].

1.2 Literature Review

There are many studies that have been done in the area of network management software. In this section, we briefly mention some of these studies:

C. S. Nimodia and S. S. Asole proposed to provide details about the network state to the administrator of network on email account and android phone, when administrator is away from office or goes out site [5].

Kim, Hyojoon and Feamster, N. focus on three problems in network management: enabling frequent changes to network conditions and state, providing support for network configuration in a high level language, and providing better visibility and control over tasks for performing network diagnosis and troubleshooting. The technologies they describe enable network operators to implement a wide range of network policies in a high-level policy language and easily determine sources of performance problems. In addition to the systems themselves, they describe various prototype deployments in campus and home networks that demonstrate how software defined networking can improve common network management tasks [3].

Yemini, Y. and Konstantinou, A.V. designed a system to replace labor-intensive configuration management with one that is automated and software-intensive. Configuration management is automated by policy scripts that access and manipulate respective network elements via a resource directory server. Resource directory server provides a uniform object-relationship model of network resources and represents consistency in terms of constraints; it supports atomicity and recovery of configuration change transactions, and mechanisms to assure consistency through changes [6].

Enck, W. and Moyer, T. designed a system as a configuration management system that attempts to address failings of current network management systems in a comprehensive and flexible way [7].

Harry Li, developed a system using simple network management protocol (SNMP) to enhance existing network management systems to meet wireless network requirements. The proposed new system can provide the following functions: discover mobile devices automatically, establish association among mobile devices and access point, monitor the wireless network performance, control access to wireless network through access point, and notify the changes of wireless network performance [8].

Liu, A.X. and Khakpour, A.R. propose a suite of algorithms for quantifying reachability based on network configurations (mainly Access Control Lists (ACLs)) as well as solutions for querying network reachability [9].

1.3 Objective

For universities today, mobility is no longer to make calls only; rather, it is considered an imperative component of their information. Mobile applications provide strong productivity. Such applications also provide more benefits across the rest of the university , such as:

1. Mobile technology has become widely used.
2. Enhanced ability for administer network.
3. Staff, analysts, and others receive information in a much more efficient and timely manner with fewer errors.
4. Top-level management receives information in real time to make better decisions.

The objective of this project was to create a mobile application on Android open source platform to assist network managers in managing their wireless networks. The application stores and display basic information about the access points and display their locations on Google maps.

SNMP protocol which provides a means to analyze the network device logs and provide statistics regarding from the access points (APs) point-of-view. However, this data is either aggregated or instantaneous information that is dependent upon the SNMP polling interval.

The application is used to conduct a measurement study and collected data is used for reporting and visualizing the congestion profile of the campus wireless network.

The advantage of this project is the use of the information of access points to monitor the wireless networks and show distribution and development wireless access points in the Eastern Mediterranean University (EMU).

1.4 Thesis Organization

Chapter 1 presents a literature review, defines the objective and describes the thesis organization. Chapter 2 describes the project tools, Android platform, Android application components, the development environment and the SNMP. Chapter 3 presents the design and implementation, system architecture, flow diagram and class diagram of the management system. Chapter 4 presents a case study with EMUWiFiManager and Chapter 5 concludes the thesis.

Chapter 2

PROJECT TOOLS

2.1 Introduction

Smart phones are used as a primary platform for information access. A very large community of people use their smart phones as one of their main communication access tools. Users are increasingly carrying their mobile devices with them almost everywhere. Mobile devices fit perfectly into an ideal environment for realizing ubiquitous computing. A main aspect of ubiquitous computing is context-aware applications where the applications collect information about the environment that the user is in and use this information to achieve their goals or improve performance [10].

This chapter describes the main project tools to create Android applications.

2.2 Android Platform

The Google Android software development kit (SDK) was released by The Open Handset Alliance in November 2007. The aim of the Android operating system is bringing more developers in mobile applications. Android is a collection of layers of programs for Android devices, containing a software system, libraries and applications. The SDK contains tools and APIs to build applications on the Android systems by using the Java programming language. Android OS is an open source platform, with adaptable development and debugging environment, which also supports a diversity of scalable user experience, which has graphics systems, multimedia support and a browser. Android enables reprocess of components and an database and reuse wireless communication tools. Android uses a Dalvik virtual machine (DVM) for Android mobile devices. On the other hand, it also supports Video, Camera, GPS, 3D applications, compass, APIs for location and map means. Developers can simply access, manage and process the free Google maps and execute on his mobile systems [11].

2.3 Android Platform Architecture

Android operating system is a stack of software components which is divided into five sections and four main layers as shown below in Figure 2.1 and Figure 2.2 [12].

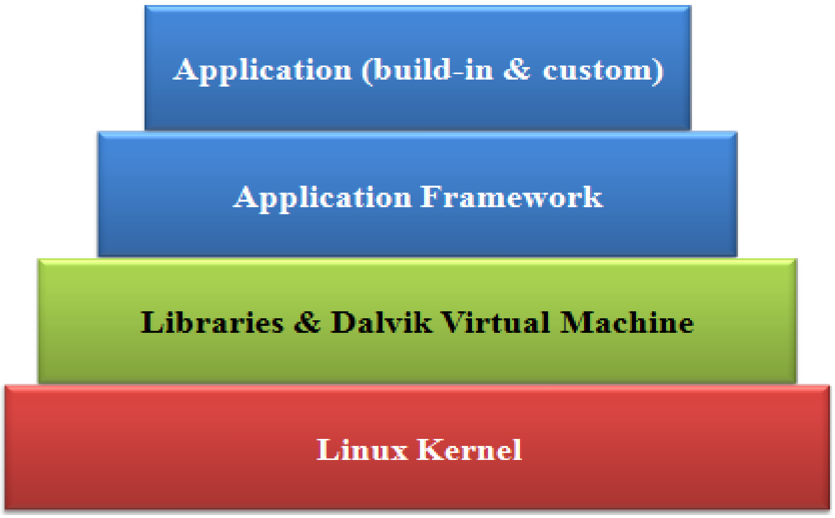

Application (build-in & custom)
Application Framework
Libraries & Dalvik Virtual Machine
Linux Kernel

Figure 2.1. Android software environment [12]

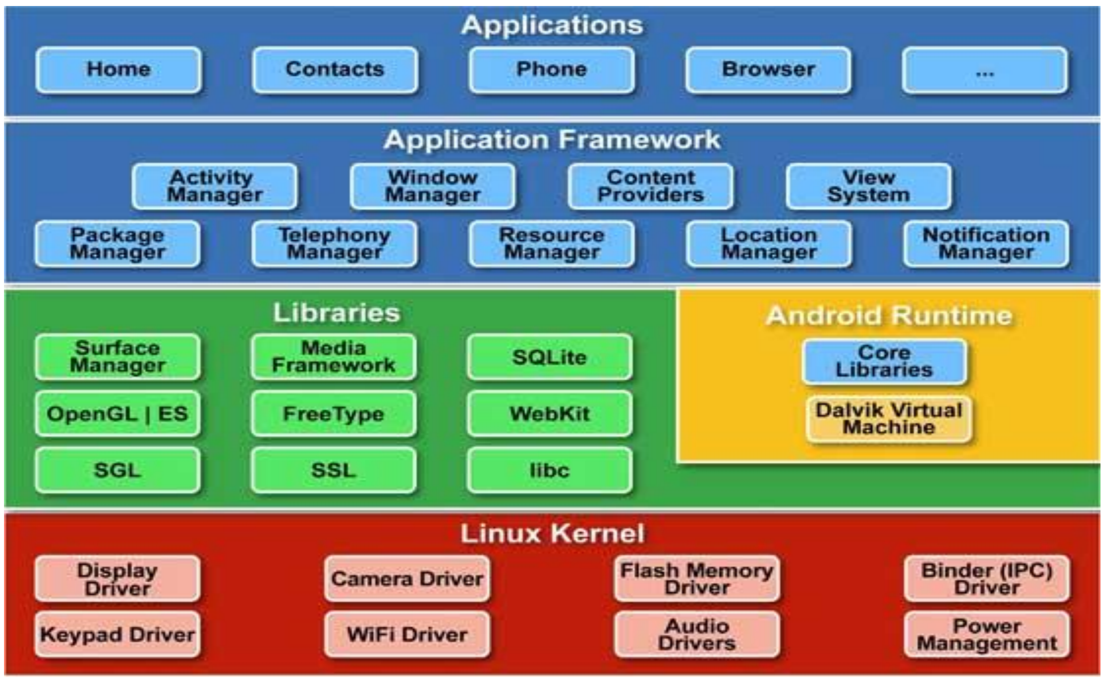


Figure 2.2. Android architecture **[13]**

Below, we simply describe these layers in more details.

Applications: The first top level layer in the framework, including as example a browser, an SMS application, a calendar, a Google map application, and contacts application. All software is Programmed using the Java language.

Application Framework: Programmers have full access to the same APIs in the application framework layer that are used by the core applications. The application architecture is designed by reusing of components. This mechanism allows every component to be replaced by the programmers. Android Application is a set of systems and services, containing a set of Activities Views to build an application, containing buttons, text views, lists, spinners, editTexts, grids, browser and even a map view which can be put into the application within some lines of code.

Libraries: Various components of the Android system use different types of libraries (as in Figure 2.2 above). These libraries are exposed to developers through the Android application.

Android Runtime: These layer containing the Dalvik Virtual Machine. Dalvik runs "dex" files, which are coverted at compile time from standard class and jar files. as shown in Figure 2.3 [13].

Linux Kernel:Android using Linux 2.6 as OS services such as process management, memory management, security, network stack, and driver model and it is a layer between the applications and all the hardware..

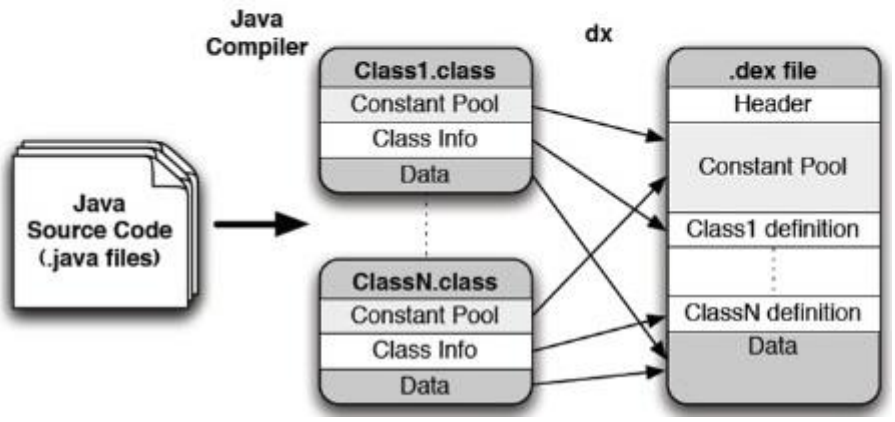


Figure 2.3. Compilation porcess for Dalvik VM applications **[13]**

2.4 Android Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact. There are following four main components that can be used within an Android application [13].

Activities: An activity is a screen with a user interface. For example, an email software might have one activity that views a list of sending emails, another activity to send an email, and another activity for showing emails. If an application has more than one activity, then one of them should be marked as the activity that is viewed when the application is launched [13].

The Android system defines a lifecycle for activities via predefined methods. The most important methods are shown in Table 2.1 [14].

Table 2.1. Important Activity lifecycle methods **[14]**

Method	**Purpose**
onCreate()	Called and startup initializations when an activity is created.
onResume()	Called when *activity* visible again.
onPause()	Called when activity gets into the foreground.. Used to release resources or save application data.
onStop()	Called when the activity is not visible to the user.

The life cycle of an activity with its most important methods is displayed in the Figure 2.4. Android has more life cycle methods but not all of these methods are guaranteed to be called. In short , Activity is a user interface that can be seen, mainly for the processing of the application as a whole works [14].

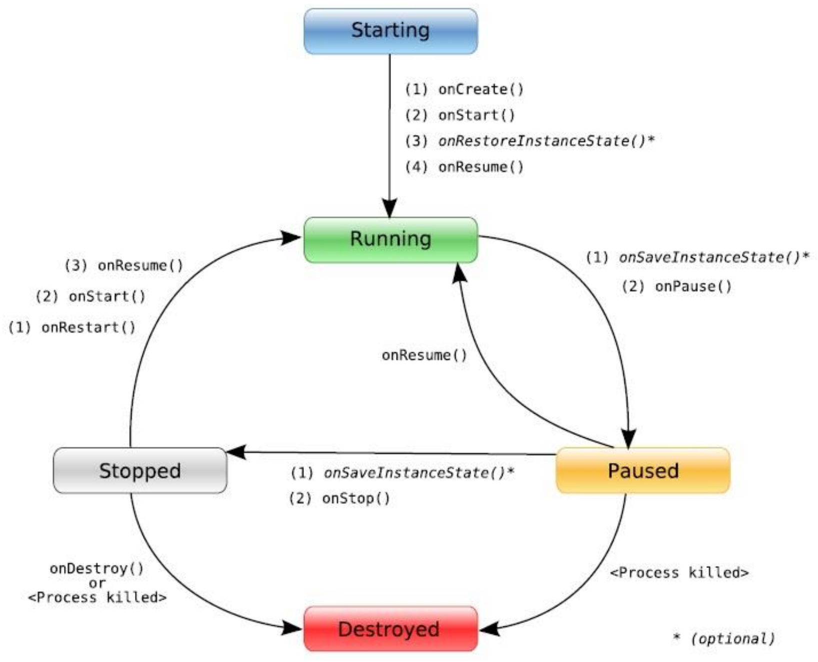


Figure 2.4. Life cycle of Android activity **[14]**

Services: A service is a component that runs in the background to perform long running operations without direct interaction with the user. For example, a service might run application in the background while the user is in a different application, or it might send or receive data over the network.

A service has lifecycle callback methods that it can implement to monitor changes in the service's state and it can perform work at the appropriate stage. The following diagram (Figure 2.5) on the left shows the lifecycle when the service is created with startService() and on the right shows the lifecycle when the service is created with bindService() [13].

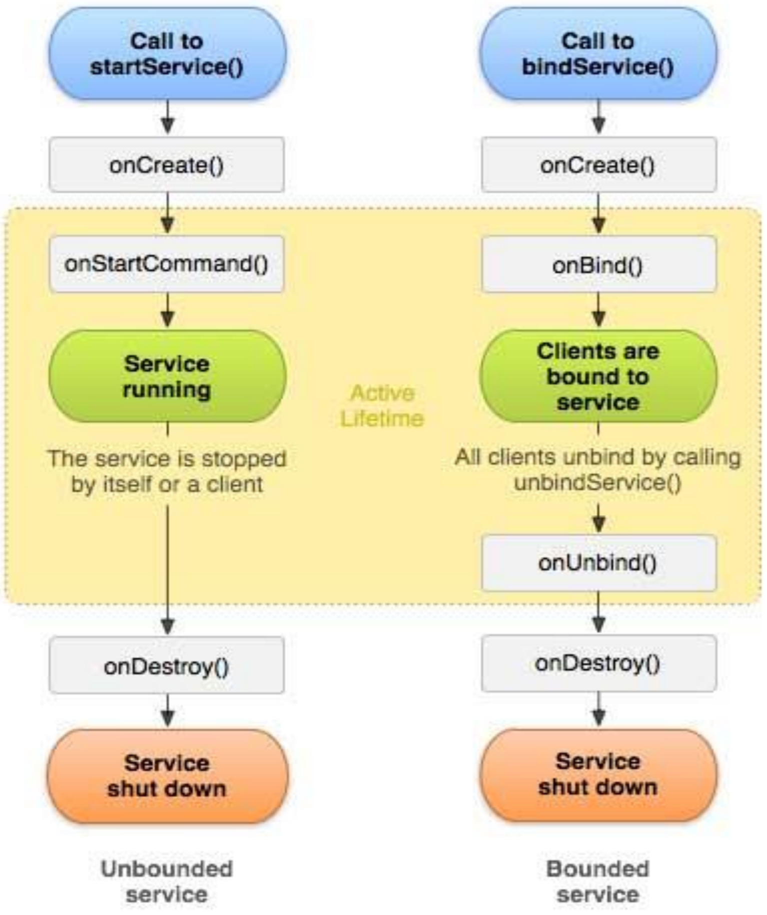


Figure 2.5. Lifecycle of an Android service **[13]**

Broadcast Receivers: Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. For example, system will send broadcast receiver when new SMS arrives or booting is done, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

Content Providers: A content provider component allows applications to access data on request. It is typically used to share data with other application. The data may be stored in the file system, the database or somewhere else entirely [13].

Additional Components: There are other components which will be used in the construction of above components. These components are [13]:

Fragments: Represents a behavior of user interface that can be placed in an Activity.

Views: UI elements that are drawn on screen including forms, lists buttons etc.

Layouts: View hierarchies that control or arranging the UI format and appearance in an activity.

Intents: Asynchronous messages which allow application components to request functionality from other components.

Resources: External elements, such as user interface strings, colors and bitmaps .

Manifest: Represents configuration file for the Android application.

2.5 Android Versions

The version history of the Android mobile OS began with the release of the Android beta in November 2007. The primary version, Android 1.0, was released in September 2008. Android is in progress development by Google and the Open Handset Alliance (OHA), and has seen a variety of updates to its base OS since its initial version [15].

Since April 2009, Android versions have been developed under a confectionery-themed code name and released consistent with alphabetical order: Cupcake (1.5), Donut (1.6), Eclair (2.0–2.1), Froyo (2.2–2.2.3), Gingerbread (2.3–2.3.7), Honeycomb (3.0–3.2.6), Ice Cream Sandwich (4.0–4.0.4), Jelly Bean (4.1–4.3), and KitKat (4.4). On 3 September 2013, Google announced that one billion activated devices now use the Android OS worldwide. The most recent major Android update was KitKat 4.4, which was released to commercial devices on 22 November 2013, via an over the air (OTA) update as show in Figure 2.6 [16].

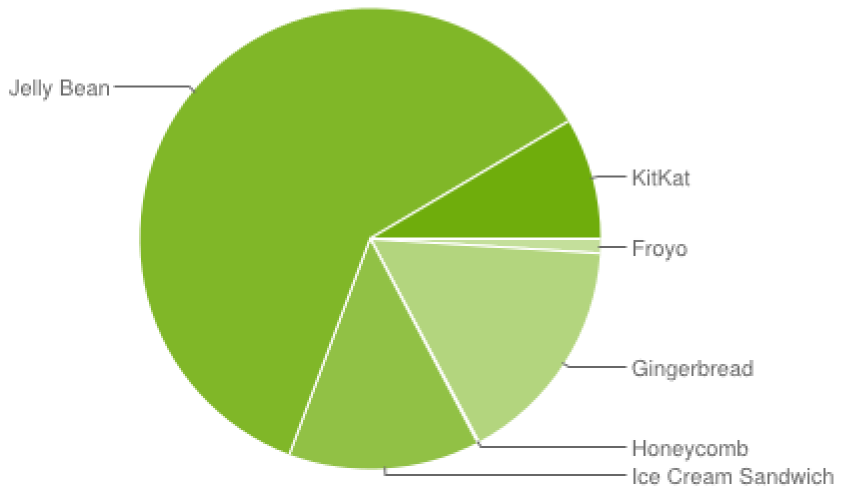


Figure 2.6. Android versions distributions April 1, 2014 **[17]**

2.6 The Development Environment

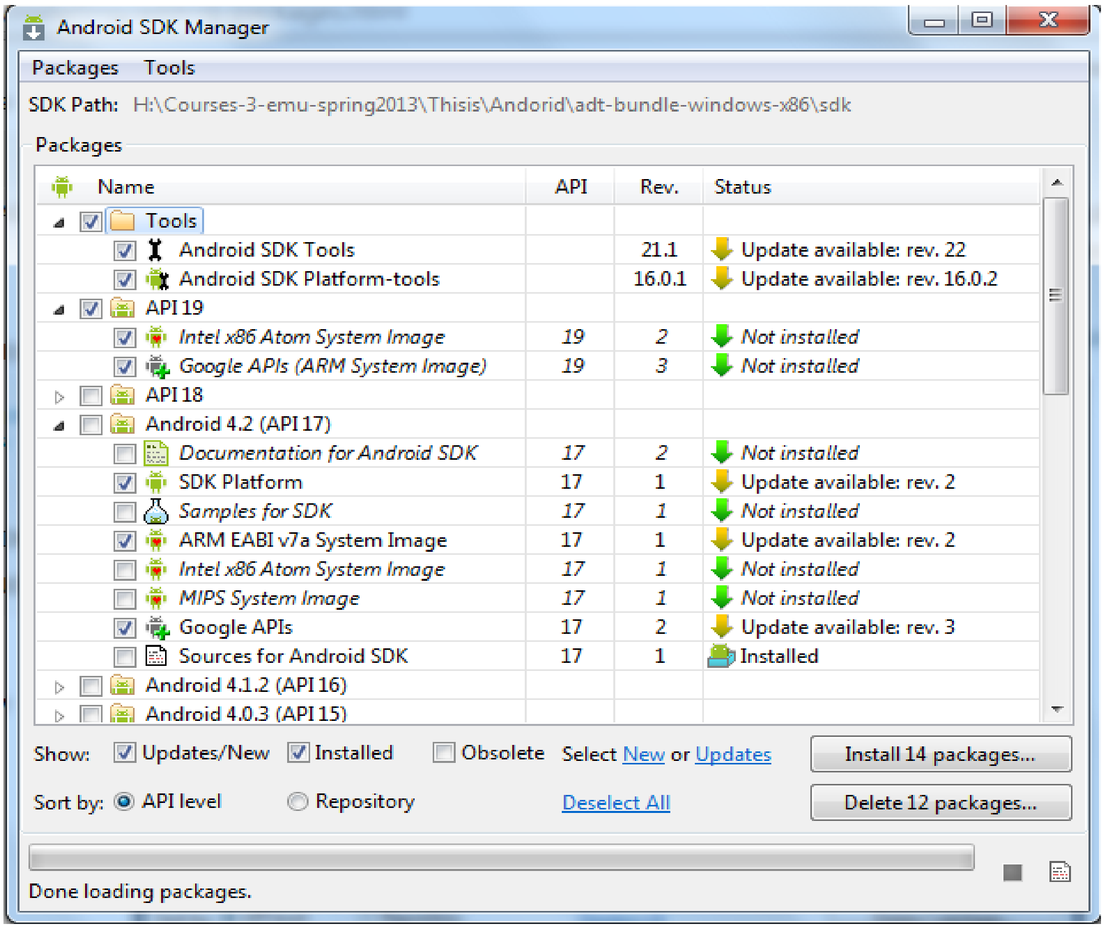
Android SDK makes use of Java programming language, similar to Java Standard Edition (J2SE), called Java Android Library. This is an advantage to programmers familiar with programming languages originating from the programming language family C. The syntax is the same as Java in terms of operands, selections, iterations, file handling. The specific Android classes and packages use different names that are not similar to Java editions, such as the View Class and the Activity Class. To develop an Android application, programmers need to make sure that the development environment has a Java version 5 or above. Java is OS independent, developer could choose developing environment OS freely [16].

2.7 Android SDK

An Android application can be developed using an Android SDK and a compatible software development environment. The Android SDK provides developers with the necessary set of tools and libraries needed to create, test and debug applications on the Android platform [17]. It is available for download on the Android official website (http://developer.android.com/sdk/index.html?hl=sk).

Although the SDK can be used to write Android programs in the command prompt, the most common method is by using an integrated development environment (IDE). The recommended IDE is Eclipse with the Android Development Tools (ADT) plug-in [17].

The Android SDK Manager shows the SDK packages that are available, installed, or which available for update as shown in Figure 2.7 [17].


Android SDK Manager

Packages Tools

SDK Path: H:\Courses-3-emu-spring2013\Thisis\Andorid\adt-bundle-windows-x86\sdk

Packages

	Name	API	Rev.	Status
✓	Tools			
✓	Android SDK Tools		21.1	Update available: rev. 22
✓	Android SDK Platform-tools		16.0.1	Update available: rev. 16.0.2
✓	API 19			
✓	*Intel x86 Atom System Image*	19	2	*Not installed*
✓	*Google APIs (ARM System Image)*	19	3	*Not installed*
	API 18			
	Android 4.2 (API 17)			
	Documentation for Android SDK	17	2	*Not installed*
✓	SDK Platform	17	1	Update available: rev. 2
	Samples for SDK	17	1	*Not installed*
✓	ARM EABI v7a System Image	17	1	Update available: rev. 2
	Intel x86 Atom System Image	17	1	*Not installed*
	MIPS System Image	17	1	*Not installed*
✓	Google APIs	17	2	Update available: rev. 3
	Sources for Android SDK	17	1	Installed
	Android 4.1.2 (API 16)			
	Android 4.0.3 (API 15)			

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates Install 14 packages...

Sort by: [✓ API level] Repository Deselect All Delete 12 packages...

Done loading packages.

Figure 2.7. Android SDK Manager

2.8 Android Emulator

Android emulator provided by the Android SDK, is a virtual mobile device that runs on the computer and enables the user to test and debug applications without a physical device as shown in Figure 2.8. The features of the emulator is defined and can be edited by the programmer using the Android Virtual Device (AVD) Manager as show in Figure 2.9 , which is a graphical UI used to configure and manage AVDs.

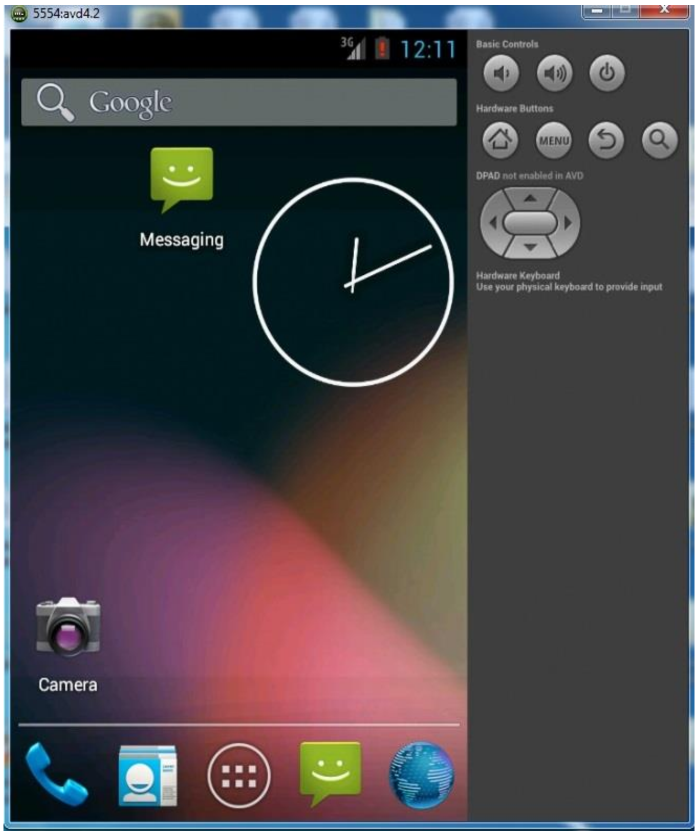
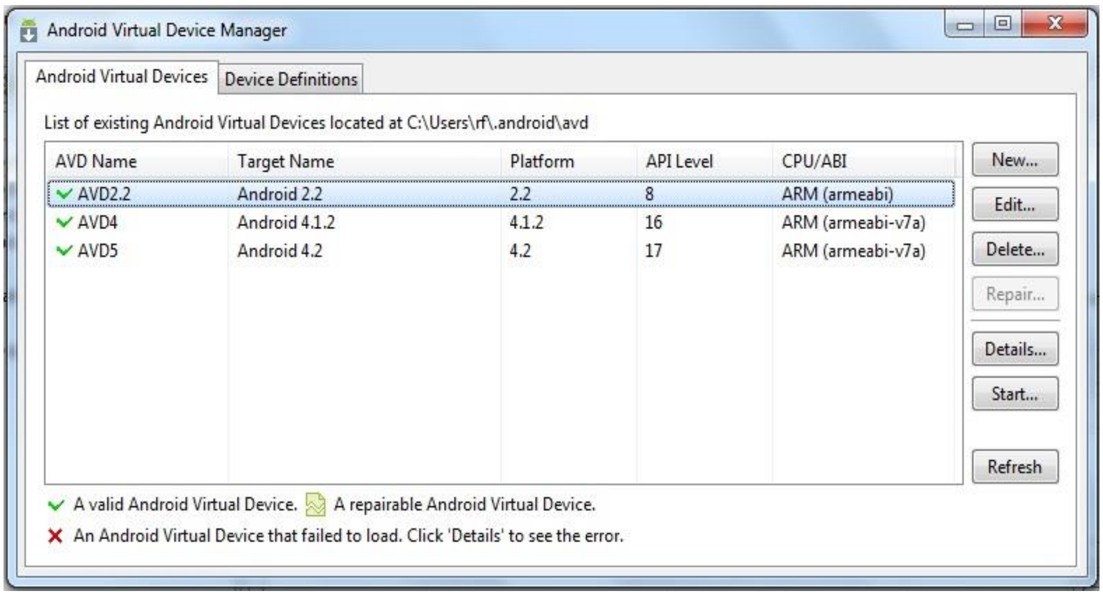


Figure 2.8. Android Emulator



Android Virtual Device Manager

Android Virtual Devices | Device Definitions

List of existing Android Virtual Devices located at C:\Users\rf\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ AVD2.2	Android 2.2	2.2	8	ARM (armeabi)
✓ AVD4	Android 4.1.2	4.1.2	16	ARM (armeabi-v7a)
✓ AVD5	Android 4.2	4.2	17	ARM (armeabi-v7a)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 2.9. AVD Manager

2.9 Google Play services

Google offers the Google Play service, a marketplace in which programmers can offer their Android applications to Android customers. Customers use the Google Play application which allows them to buy applications from the Google Play service [17].

Google Play services also provides an update service. If a programmer uploads a new version of his application to Google Play, this service notifies users that an update is available and allows them to install the update [17].

Google Play provides access to libraries for Android application programmers. For example, it provides a library to use and display Google Maps and another to synchronize the application state between different Android installations. These services have the advantage that they are available for older Android versions and can be updated by Google without the need for an update of the Android version on the smart phone [17].

2.10 Eclipse

In computer programming, *Eclipse* is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages C, Ada , C++, Fortran, COBOL, Haskell, JavaScript , Perl, Lasso, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Scheme, Groovy, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse CDT for C/C++, the Eclipse Java development tools (JDT) for Java and Scala, and Eclipse PDT for PHP, among others. The initial codebase was released by IBM . The Eclipse SDK, which includes the Java development tools, is meant for Java programmers. Users can extend its abilities by installing plug-in development toolkits written for the Eclipse Platform and can write and contribute their own plug-in modules [18].

2.11 Java

Java is a class-based computer programming language. It is general-purpose, structured and generic. Android applications are written in the Java Programming language. An Android application is highly based on Java fundamentals. Java incorporates with several powerful features and libraries of many powerful programming languages like C, C++. The reasons for picking Java as a basic programming language for Android application are [19]:

1. It is easy to learn
2. It is independent of OS
3. It is an object oriented programming language
4. Java line code is compiled and run by Virtual Machine

2.12 Extensible Markup Language (XML)

XML is a markup language. It contains some of the very simple, scalable, and flexible text format that is both machine-readable and human-readable. It Defines documents with a standard format that can be read by any XML-compatible application. XML is a commonly used data format on the Internet. XML is easy to compile and manipulate programmatically. Android resources preprocess the XML into the compressed binary format and stores it on the device. Most of the UI layout, screen contents are declared in XML files [20].

2.13 SQLite Database

SQLite is an in-process library that implements a serverless, self-contained, transactional database engine and zero-configuration. It is free to use for any purpose. It is compact and lightweight hence it is easily deployable to any system. It is supported by the many Linux and Windows operating systems and can be ported easily to other systems. The data types supported are TEXT (to hold string values), INTEGER (to hold integer values) and REAL (to hold precision floating-point values). Android provides the SQLite database to allow for data storage in an application. An application in the Android system may have a private database and this can only be accessed and managed within the application code [21].

2.14 Wi-Fi APIs

The Wi-Fi APIs provide classes by which applications can communicate with the hardware wireless stack that provides Wi-Fi network access. Almost all information from the device is available, including the negotiation state, link speed, IP address and more, plus information about other networks that are available. Some other API means include the ability to initiate Wi-Fi, scan, save, add and terminate connections [22].

2.15 Google Maps Android API

Google Maps is a Web-based service that provides information about sites around the world and geographical regions. Besides conventional road maps, Google Maps provides satellite views and aerial of many places.

The Google Maps APIs make it possible for Web site administrators to put Google Maps into a custom site such as a community service page.

Google Maps for Mobile provides a location service that utilizes the Global Positioning System (GPS) location of the mobile device along with data from cellular networks and wireless LANs [23].

2.16 Google Locations Services API

Most Android devices allow to determine the current GPS. This can be done via a GPS module, via cell tower triangulation or via wireless LANs [23].

The Google Location Services API, part of Google Play Services, offers a powerful tools, high-level framework such as location provider choice and power management. Location Services also offers new tools such as activity detection that aren't available in the framework API. Programmers who are using the framework API, as well as programmers who are just now adding location-awareness to their applications, should firmly consider using the Location Services API [23].

2.17 SNMP

Simple Network Management Protocol (SNMP) is used in network management systems to monitor network devices for cases that require administrate. SNMP gives management data in the form of variables on the managed systems. These variables may then be queried or set by managing applications. SNMP itself does not define which variables are reachable. Alternatively, SNMP uses an extensible design, where the available information is defined by *Management Information Bases* (MIBs) that are often owned by individual vendors. MIBs describe the structure of the management data of a device subsystem in a hierarchical namespace containing *Object Identifiers* (OID). Each OID identifies a variable that can be read or set via SNMP [24].

SNMP was introduced in 1988 to meet the growing need for a standard for managing *Internet Protocol* (IP) devices. The main core of SNMP is a simple set of operations that provides the ability to query and set the state of some devices to network administrators. Although SNMP is capable of managing a wide variety of network devices such as printers, personal computers, servers, power supplies, etc., it is typically associated with routers and other network devices. Just as with other protocols, SNMP is defined by The *Internet Engineering Task Force* (IETF) using *Request for Comments* (RFC) specifications. Currently [24], SNMP is composed of:

1. SNMP Manager
2. SNMP agent.
3. Managed network devices and
4. Database of MIB.

2.17.1 SNMP Manager

SNMP manager or management interacts with SNMP agents that are active in the network devicess. It is a computer to run one or more SNMP manager. The main functions of SNMP Manager are to:

1. Query agents
2. Get requested values from agents
3. Change values of variables in the agents
4. Asynchronous with SNMP agents with in the network.

2.17.2 Managed Network Devices

A managed network devices as shown in Figure 2.10 such as routers, switches, access points, servers, workstations, printers requires some form of monitoring and management [25].

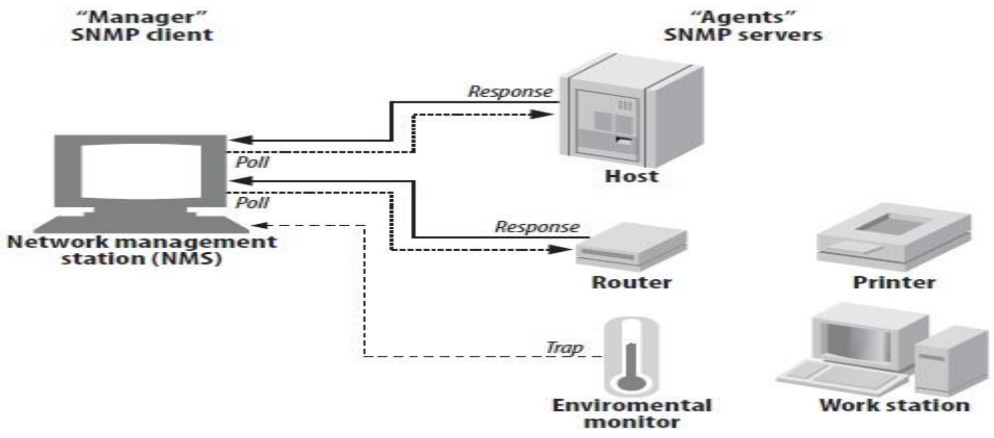


Figure 2.10. SNMP manager and agents **[25]**

2.17.3 SNMP Agents

SNMP agent is a software installed in the network device. It collects the information from the device and give it to the SNMP manager. These agents can be general such as Net-SNMP or vendor specific such as HP insight [25]. The main functions of SNMP agent are:

1. Collects data for its local environment
2. Retrieves and stores data as defined in the MIB.
3. Sends an occurrence to the manager.

SNMP agent perpetuates a management data describing the managed system parameters. It exposes an information database in the form of variables on the managed systems, which describe the system configuration on the agent. These variables can then be queried (and sometimes set) by managing applications [25].

2.17.4 Management Information Base and Object ID

SNMP uses an extensible design, where the available information is defined by management information bases (MIBs). MIBs describe the structure of the management data of a device subsystem; they use a hierarchical namespace containing object identifiers (OID). Each OID is unique and identifies a variable that can be read or set via SNMP [25].

MIBs is divided into two types:

Scalar: defines a single object instance, the result can be only one.

Tabular: objects define multiple related object instances grouped in MIB tables..

Absolute OIDs specify a path to an attribute from the root of the OID tree. Absolute OID names always begin with a dot and must specify every node of the OID tree from the top-most node to the specific managed object. For example, "sysDescr" is .1.3.6.1.2.1.1.1 as shown in Figure 2.11 [25].

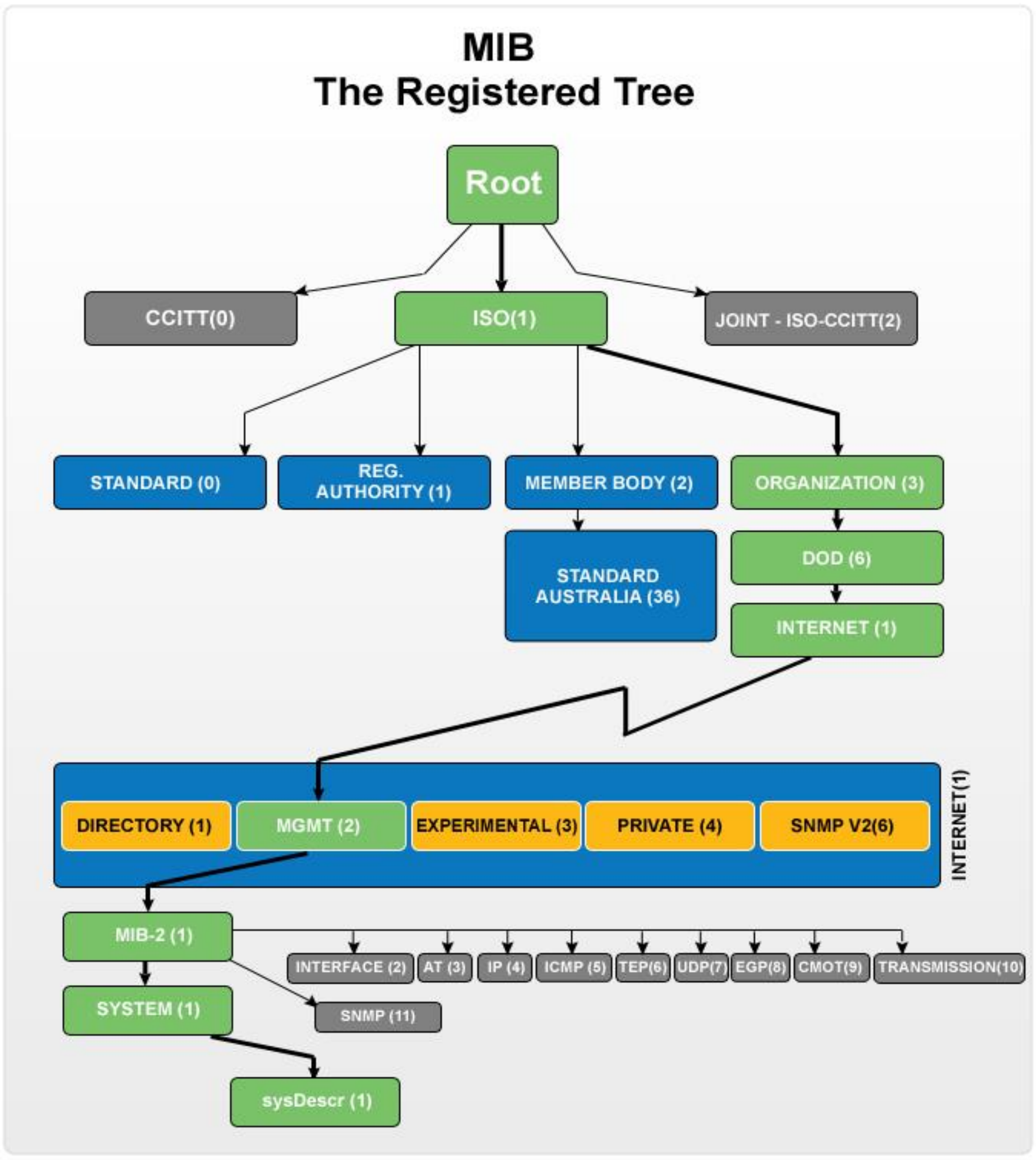
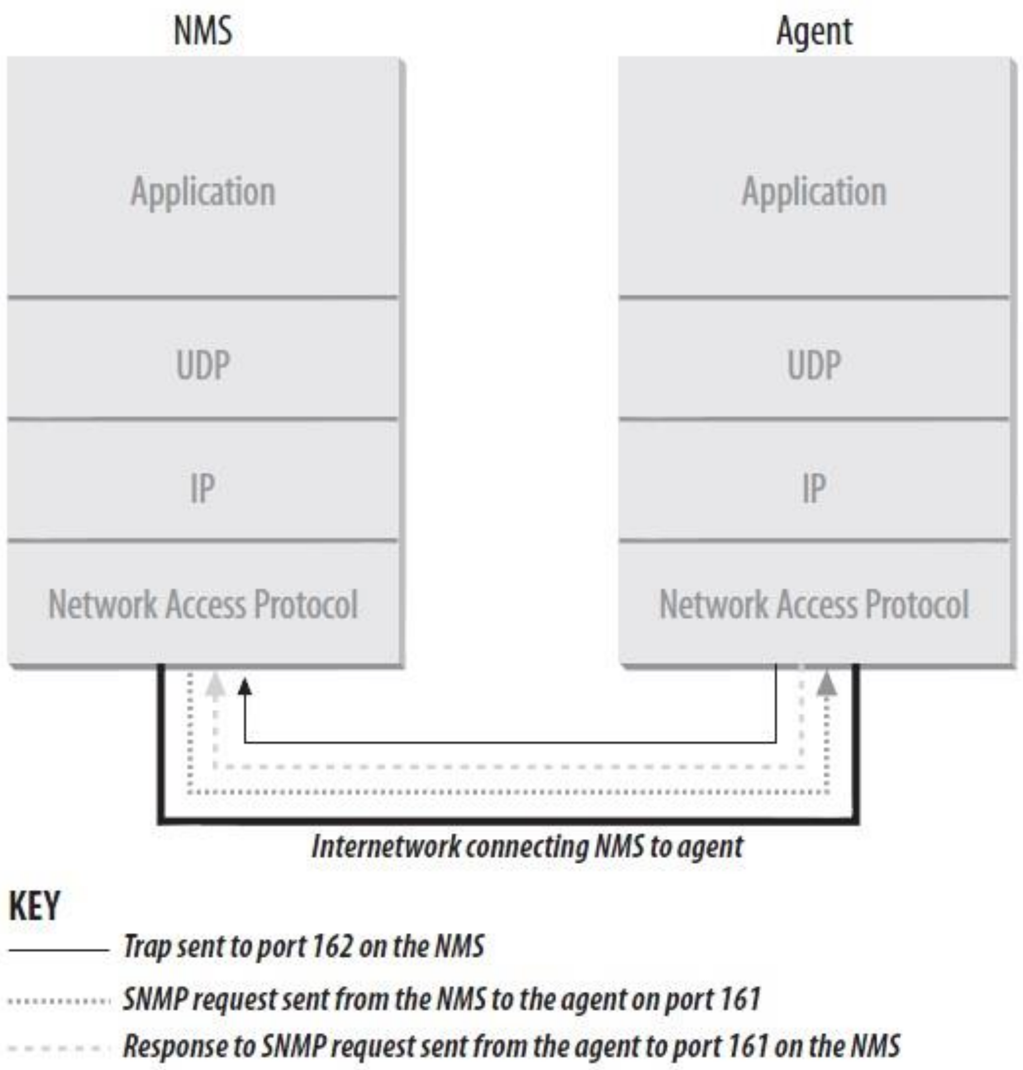


Figure 2.11. MIB Tree Diagram [24]

2.17.5 SNMP Communication

Figure 2.12 shows the TCP/IP protocol suite, which is the basis for all TCP/IP communication. Any device that wishes to communicate on the Internet (e.g., Windows systems, Unix servers, Cisco routers, etc.) must use this protocol suite. This model is often referred to as a protocol stack since each layer uses the information from the layer directly below it and provides a service to the layer directly above it [24].



NMS
Agent
Application
UDP
IP
Network Access Protocol
Application
UDP
IP
Network Access Protocol
Internetwork connecting NMS to agent
KEY
Trap sent to port 162 on the NMS
SNMP request sent from the NMS to the agent on port 161
Response to SNMP request sent from the agent to port 161 on the NMS

Figure 2.12. TCP/IP communication model and SNMP **[24]**

2.17.6 Versions of SNMP

SNMP has gone through important updates. Nonetheless SNMP v1 and v2c are the most implemented versions of SNMP. SNMP v3 has recently started and it is more secure [24].

*SNMP version 1:*the first version of the protocol, easy to set up – only requires a plaintext community.

*SNMP version 2c:*This is the update protocol, which includes improvements of SNMP version 1 in the protocol packet types, transport mappings and MIB structure elements.

*SNMP version3:*The evolution of SNMP version 3 was based on the security issues. SNMPv3 facilitates remote configuration of the SNMP entities.

Nonetheless each version had developed towards rich functionalities;the security aspect on each update was given as additional emphasis. Table 2.2 shows security aspect for each editions [24].

Table 2.2. SNMP security aspect **[24]**

SNMP Version	**Security Aspect**
v1	Community security
v2c	Community security
v2u	User security
v2	Party security
v3	User security

Chapter 3

SOFTWARE DESIGN AND IMPLEMENTATION

This chapter presents the design of the mobile application whose purpose is to meet the objective defined in Chapter 1.

3.1 Requirements

The aim of this project was to create a mobile application to assist network managers for reporting and visualizing the congestion profile of their campus wireless network. So the main requirements are:

1.Storing the information of APs for selected site on mobile.

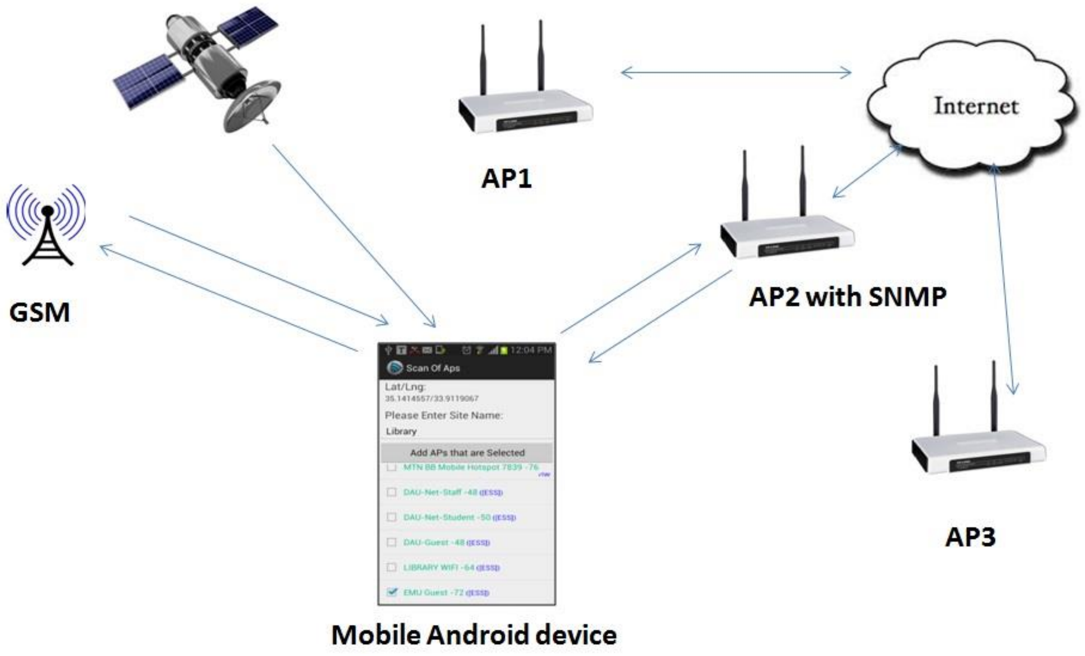
2.Putting the locations of these APs on Google Maps.

3. Using SNMP to analyze the network device logs.

3.2 Design Methodology

3.2.1 System Architecture

The system architecture has been illustrated in Figure 3.1. An Android device (not necessarily a smartphone, it could be a tablet) with the EMUWiFiManager application already installed communicates with the SNMP of AP that uses a special SNMP API .



12:04 PM
Scan Of Aps
Lat/Lng:
35.1414557/33.9119067
Please Enter Site Name:
Library
Add APs that are Selected
- [ ] MTN BB Mobile Hotspot 7839 -76
- [ ] DAU-Net-Staff -48 ([ESS])
- [ ] DAU-Net-Student -50 ([ESS])
- [ ] DAU-Guest -48 ([ESS])
- [ ] LIBRARY WIFI -64 ([ESS])
- [x] EMU Guest -72 ([ESS])

Figure 3.1. System Architecture

On the other hand, the application uses all available techniques (satellite, cell towers, Wi-Fi) to inquire about the current location and displays it on Google Maps.

3.2.2 Flow Diagram

A flow diagram showing how users can run this project is shown in Figure 3.2 below.

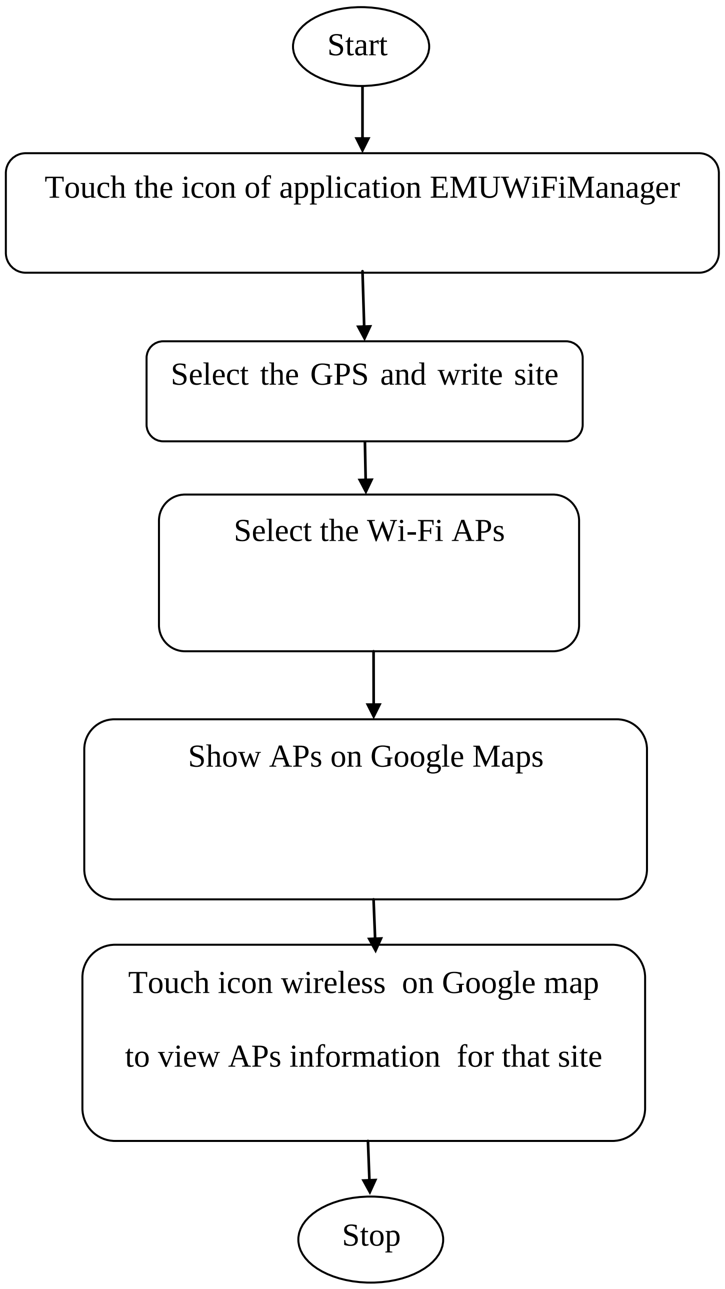

Start
Touch the icon of application EMUWiFiManager
Select the GPS and write site
Select the Wi-Fi APs
Show APs on Google Maps
Touch icon wireless on Google map to view APs information for that site
Stop

Figure 3.2. Flow Diagram

3.2.3 Data Flow diagram

The data flow diagram, is a simple graphical design that shows the flow of information that can be used in the system in terms of the inputs data and the outputs.

The figure below (Figure 3.3) shows the data flow diagram for user module. The data flow starts with adding new site by scanning for available APs to that site and retrieving location point, site name, select APs and then adding all these information to SQLite database. User can list sites and delete any AP from site and view this AP on Google Maps and view SNMP tree. Finally, user can exit from application.

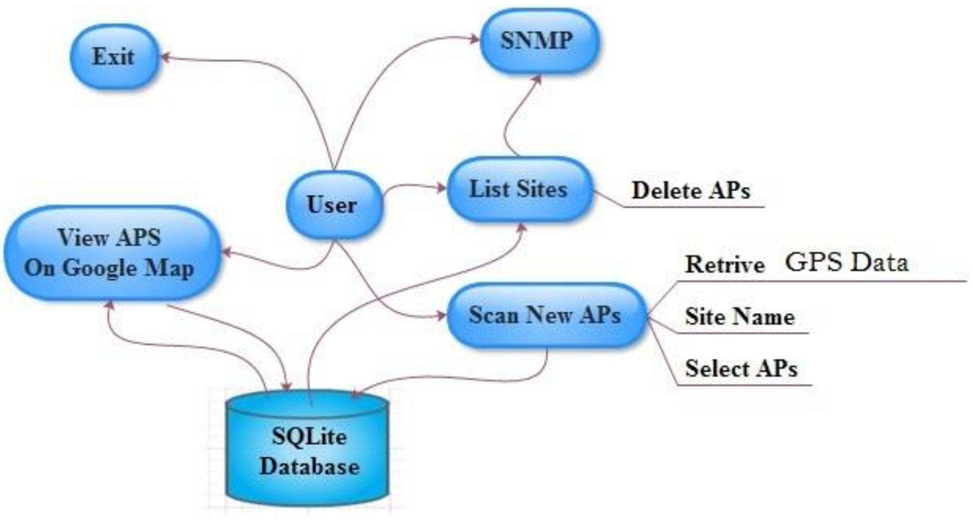


Figure 3.3. Data Flow Diagram

By using the SQLite class, we created a database "myDB1" and table "Accesspoints" in this database to store information about APs. The structure of this table is shown in the Table 3.1.

Table 3.1. Accespoints Database Table Structure

Field	**Type**	**Key**
apid	INTEGER	PRIMARY
Ssid	VARCHAR	
Mac	VARCHAR	
Wep	VARCHAR	
Lat	VARCHAR	
Lng	VARCHAR	
Dep	VARCHAR	
Dept	VARCHAR	
Apdate	VARCHAR	
Snr	VARCHAR	
IP	VARCHAR	

3.3 Development Overview And Components

The mobile application was developed in the Eclipse environment using the Android software development kit. This project (EMUWiFiManager) uses google-play-services_lib which contain all API-related library for Google maps.

The allclassess.emuwifimanager package contains all the interface-related classes. Figure 3.4 below shows the structure of the project in the Eclipse software.

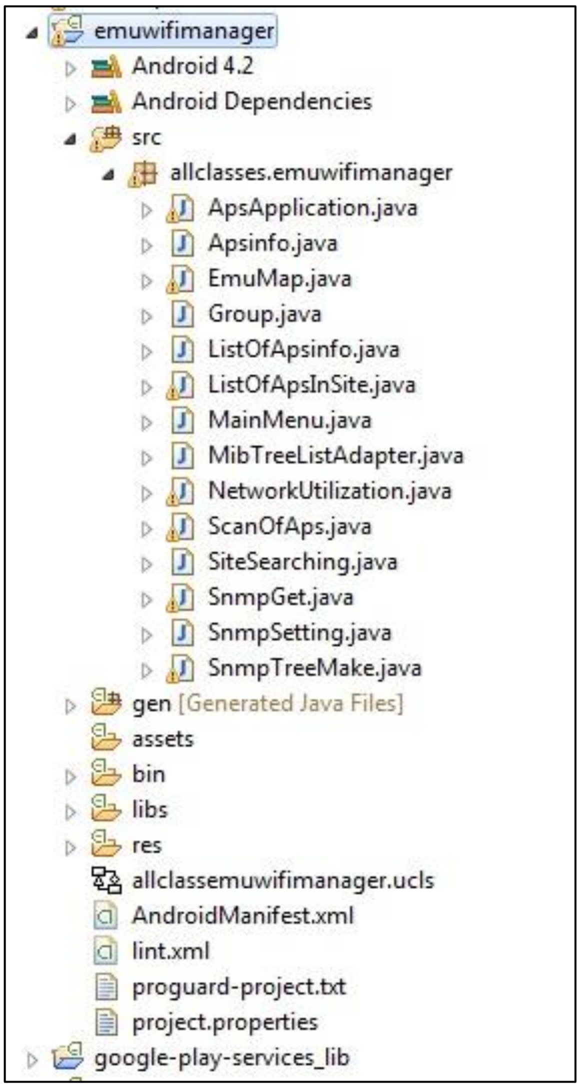
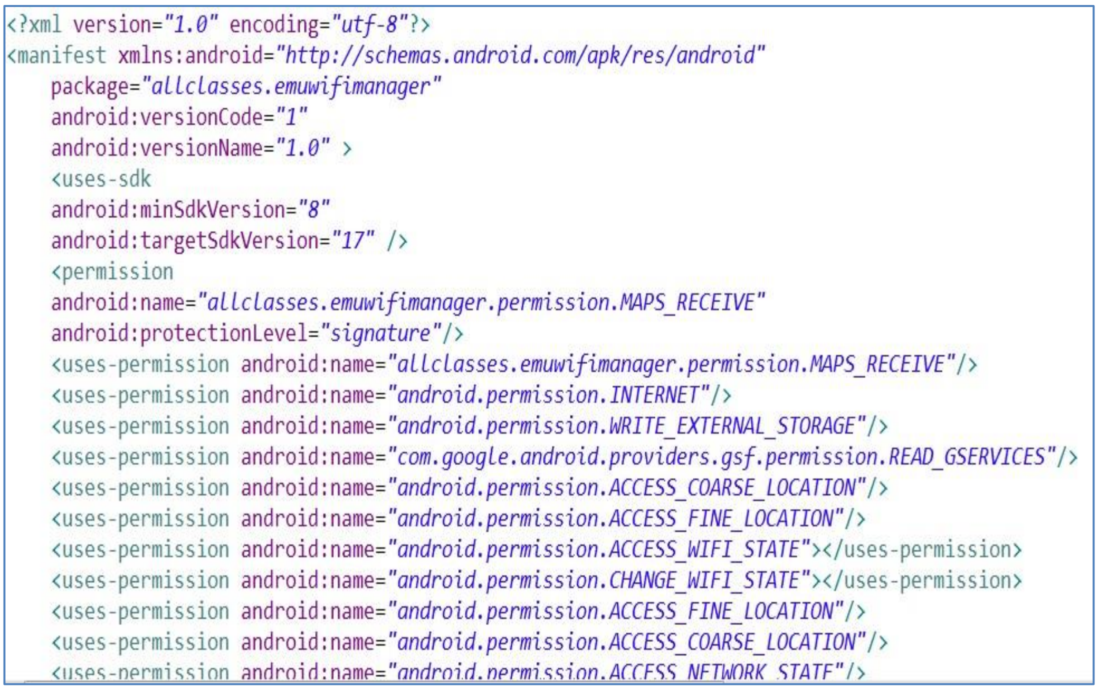

```
emuwifimanager
    Android 4.2
    Android Dependencies
    src
        allclasses.emuwifimanager
            ApsApplication.java
            Apsinfo.java
            EmuMap.java
            Group.java
            ListOfApsinfo.java
            ListOfApsInSite.java
            MainMenu.java
            MibTreeListAdapter.java
            NetworkUtilization.java
            ScanOfAps.java
            SiteSearching.java
            SnmpGet.java
            SnmpSetting.java
            SnmpTreeMake.java
    gen [Generated Java Files]
    assets
    bin
    libs
    res
    allclassemuwifimanager.ucls
    AndroidManifest.xml
    lint.xml
    proguard-project.txt
    project.properties
google-play-services_lib
```

Figure 3.4. Project Components Structure in Eclipse

The application is configured to a minimum API level of 8 and declares permissions to use the maps, Wi-Fi, internet, storage, location and network functionalities of the Android system. Figure 3.5 below shows the configuration of the *AndroidManifest.xml* file.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="allclasses.emuwifimanager"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17" />
    <permission
    android:name="allclasses.emuwifimanager.permission.MAPS_RECEIVE"
    android:protectionLevel="signature"/>
    <uses-permission android:name="allclasses.emuwifimanager.permission.MAPS_RECEIVE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Figure 3.5. AndroidManifest.xml file

3.4 Class Diagrams and Activities

In this part , we will briefly describe the class diagrams overview and its activities in the software implementation in Eclipse and execution on Android. The class and its relationships described by Unified Modeling Language (UML) as shown in Figure 3.6.

3.4.1 Mainmenu Class Activity

Figure 3.7 shows us the GUI activity which provides main options to user to interact with project, and the code of class is included in appendix A.

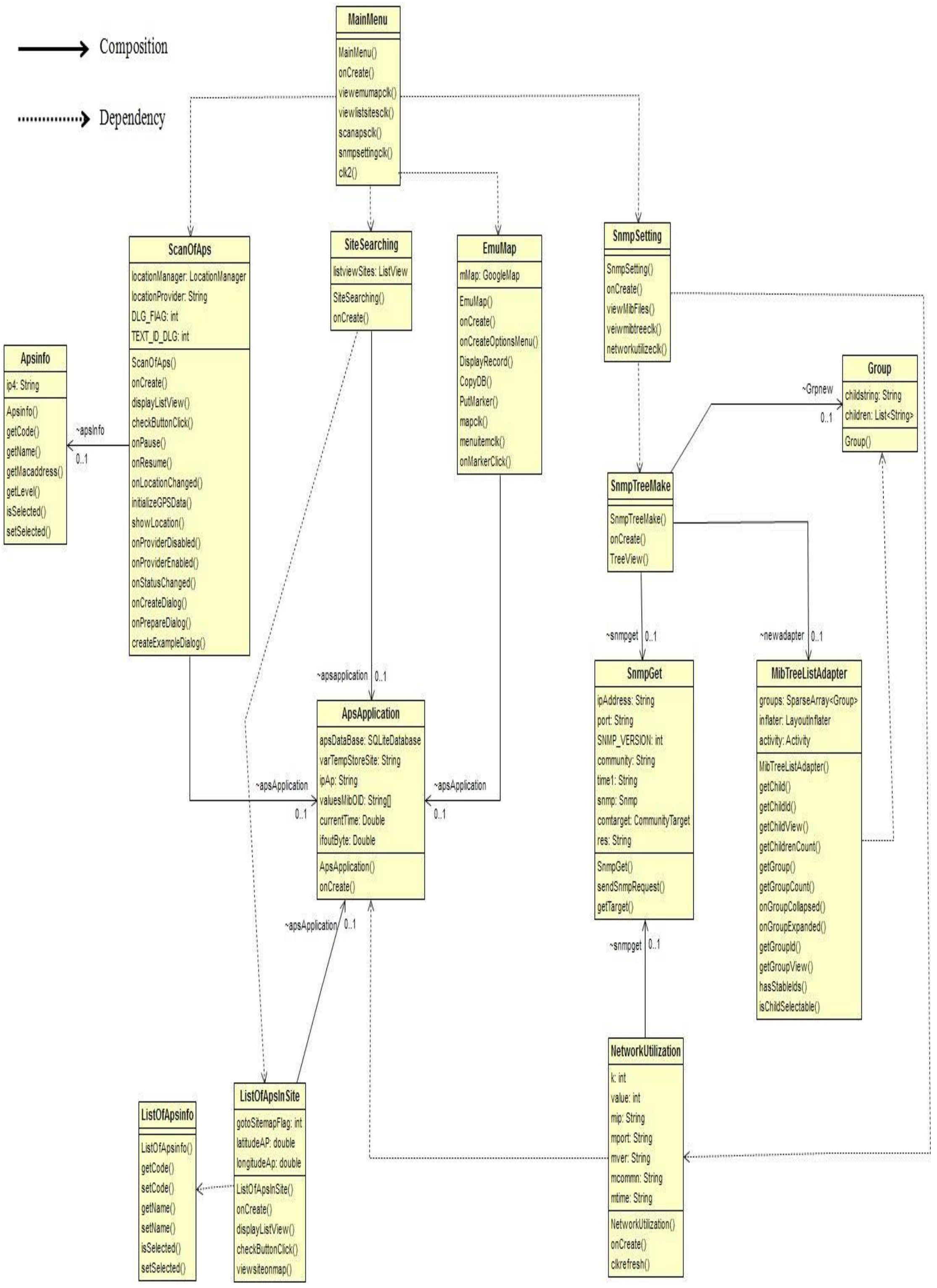


Figure 3.6. Unified Modeling Language (UML)

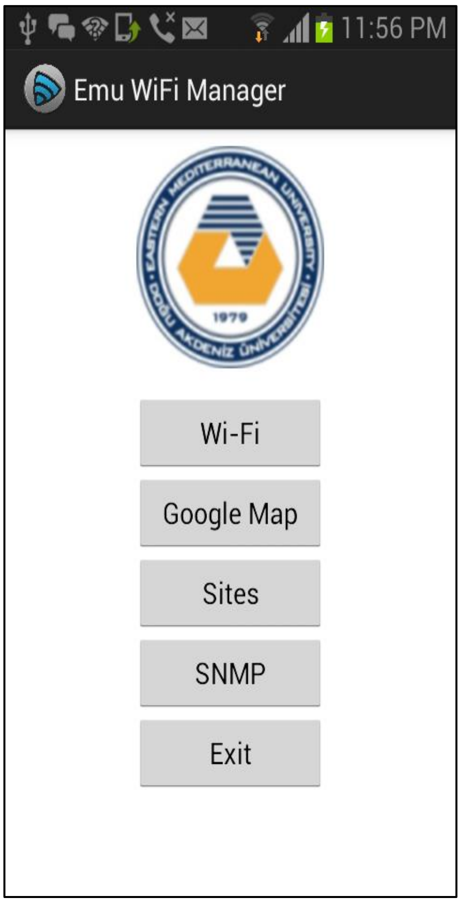


Figure 3.7. Mainmenu Activity

3.4.2 ScanOfAPs Class Activity

ScanOfAPs activity is designed to scan and view all APs in the range. Also this activity can enable Wi-Fi on Android device if it is disabled.

As shown in sequence diagram (Figure 3.8), a user starts the scanning process by touching the Wi-Fi button which initiates the ScanOfAPs class.

Finally, the user must enter the site name and select the APs related to this site and then touch on Add APs button to store this AP in the SQLite database of Android device as shown in Figure 3.9 and the code of class is included in Appendix B.

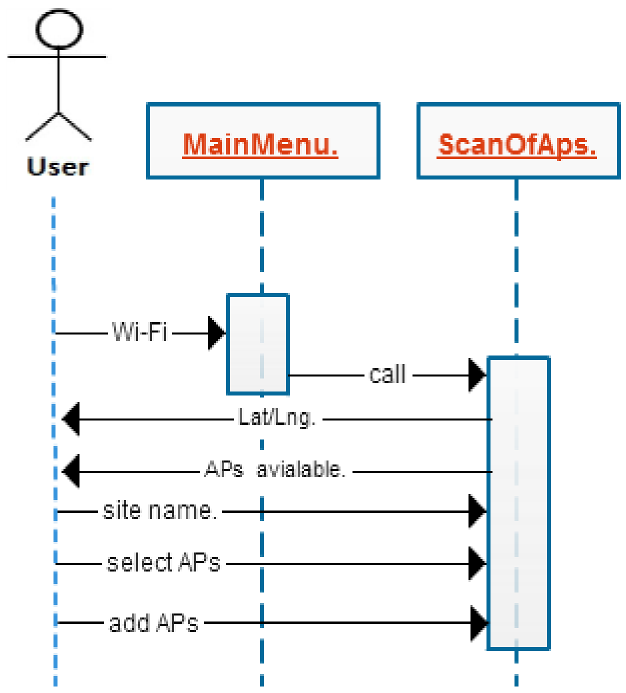
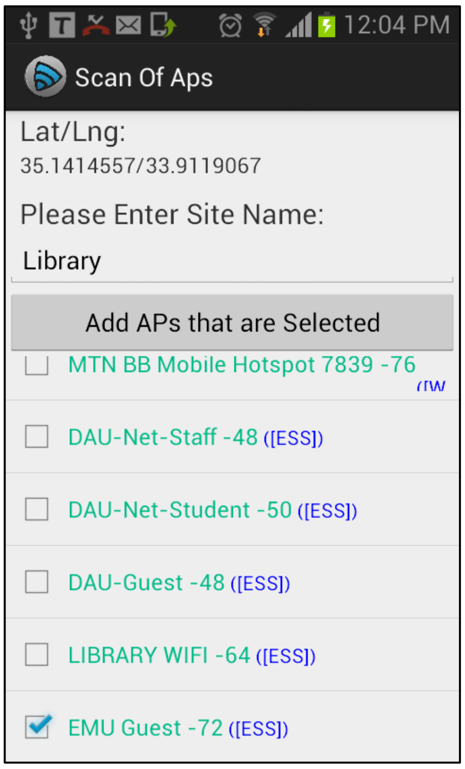


Figure 3.8. Sequence Diagram of ScanOfAPs activity



Lat/Lng:
35.1414557/33.9119067

Please Enter Site Name:
Library

Add APs that are Selected

☐ MTN BB Mobile Hotspot 7839 -76 ([W

☐ DAU-Net-Staff -48 ([ESS])

☐ DAU-Net-Student -50 ([ESS])

☐ DAU-Guest -48 ([ESS])

☐ LIBRARY WIFI -64 ([ESS])

☑ EMU Guest -72 ([ESS])

Figure 3.9. ScanOfAPs activity

3.4.3 EMUMap Class Activity

This activity is designed to view icons of wireless tower markers on Google Maps for the sites selected ot EMU. As shown in sequences diagram (Figure 3.10) we can change the map into three types : normal , hybrid and satellite as shown in Figures 3.11 , 3.12 and 3.13 respectively below. By touching this icon, the ListOfApsInSite activity well display the list of APs in that site as shown in Figure 3.14 and the code of class is included in Appendix C.

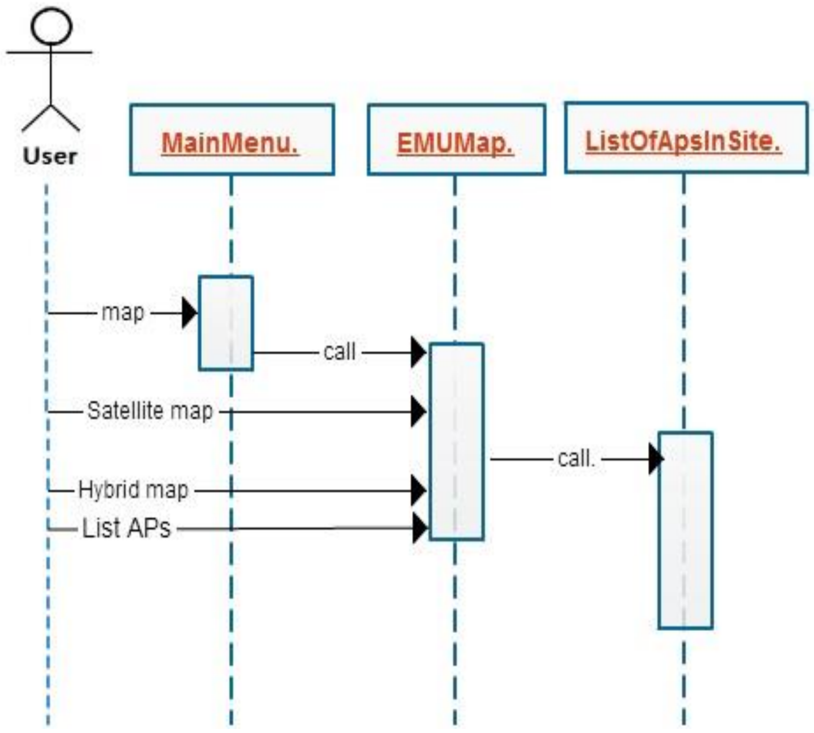


Figure 3.10. Sequence Diagram of EMUMap activity

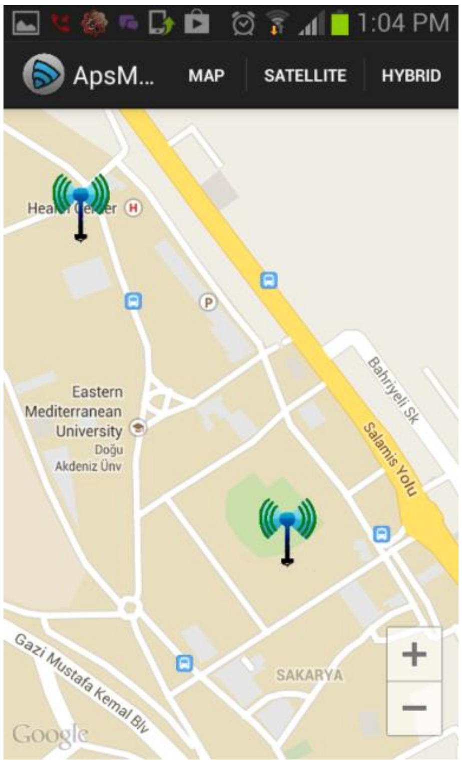


Figure 3.11. Noraml map

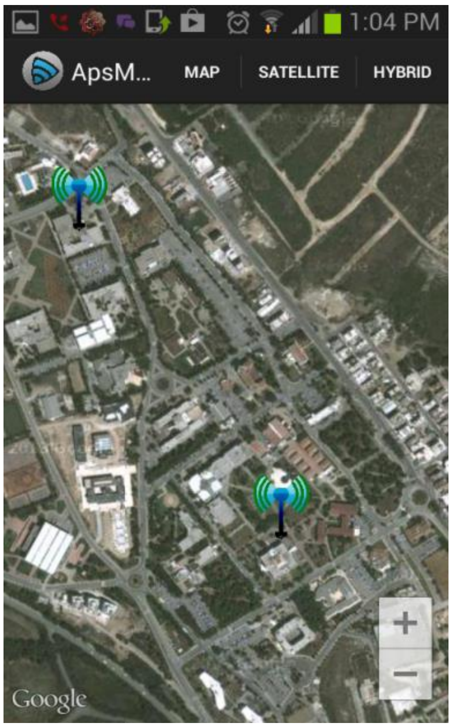


Figure 3.12. Satellite map

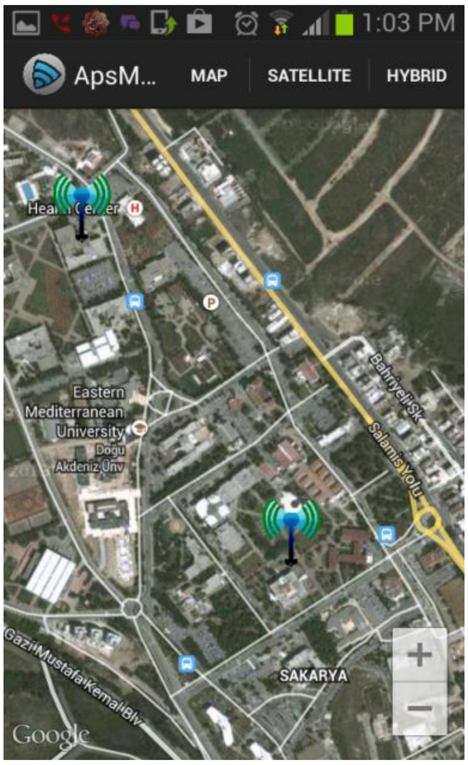
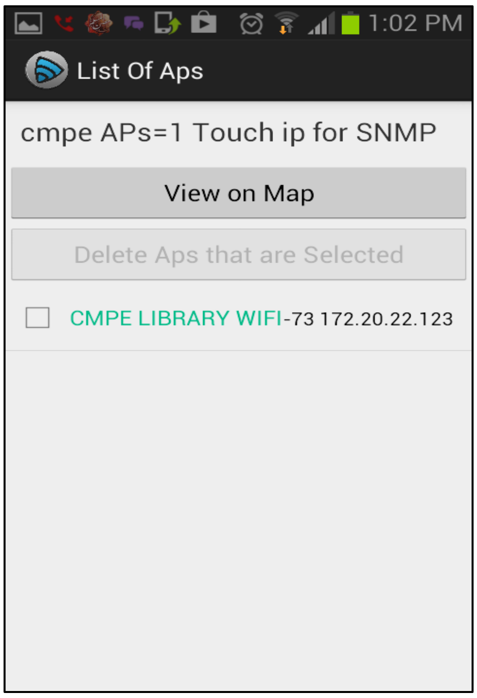


Figure 3.13. Hybrid map



cmpe APs=1 Touch ip for SNMP

View on Map

Delete Aps that are Selected

CMPE LIBRARY WIFI-73 172.20.22.123

Figure 3.14. ListOfApsInSite activity

3.4.4 SiteSearching Class Activity

The SiteSearching activity adds search functionality to the application which can filter the list sites names with a matching string, hence it provides user an easy way to find the information he needs as shown in sequence (Figure 3.15).The SiteSearching activity (Figure 3.16) and the code of class is included in Appendix D.

3.4.5 ListOfApsInSite Class Activity

As we show in Figure 3.14, the ListOfApsInSite activity can return us to EmuMap activity to show the site map with some zooming by clicking on (view on map) button as shown in Figure 3.17.

On the other hand, the ListOfApsInSite activity can going us to SnmpSetting activity by touching the IP address in front of AP and the code of class is included in Appendix E.

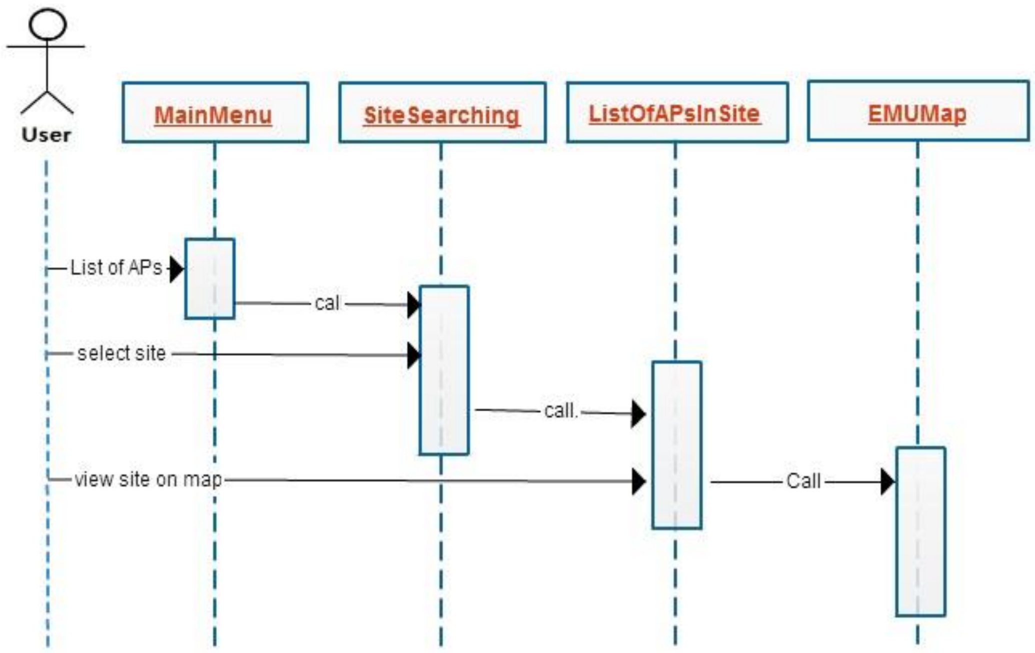


Figure 3.15. Sequence Diagram of SiteSearching activity

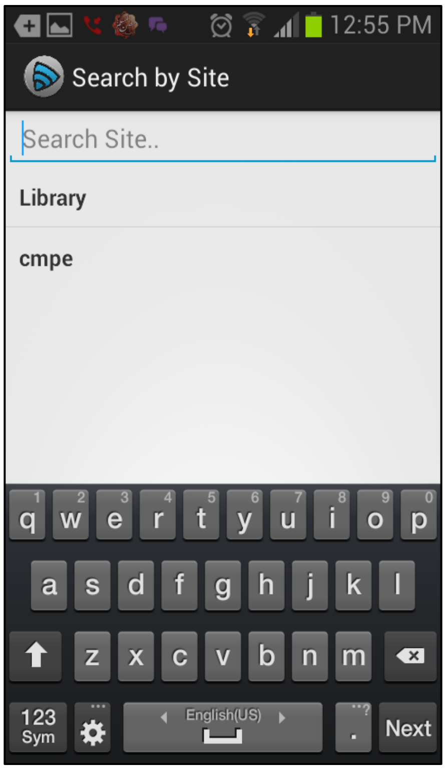


Figure 3.16. SiteSearching activity

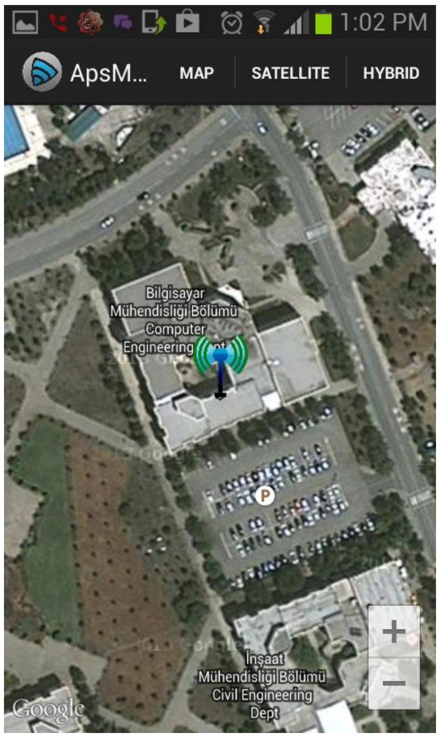

1:02 PM
ApsM...
MAP
SATELLITE
HYBRID
Bilgisayar Mühendisliği Bölümü Computer Engineering
İnşaat Mühendisliği Bölümü Civil Engineering Dept
Google

Figure 3.17. EmuMap activity

3.4.6 SnmpSetting Class Activity

SnmpSetting activity is designed to setting up SNMP protocol .The user (as shown in sequence diagram Figure 3.18) must enter the parameters (mib file name,ip,port, version,community and time out) as shown in Figure 3.19.By touching on MIB file default name, the dialog (Figure 3.20) is shown and we can select another MIB file and the code of class is included in Appendix F.

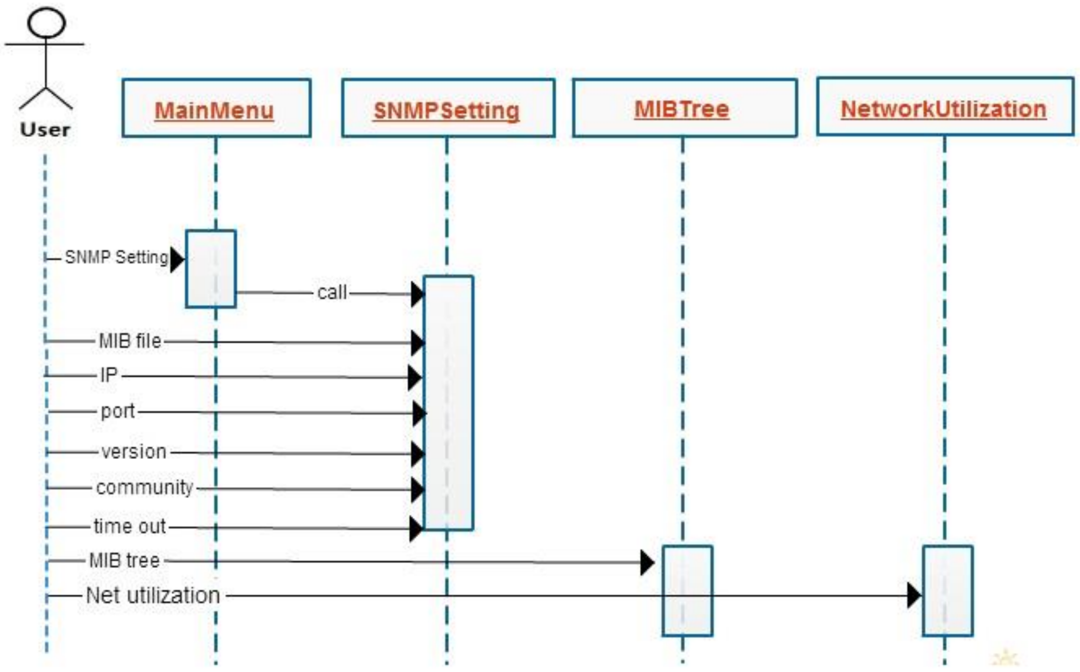
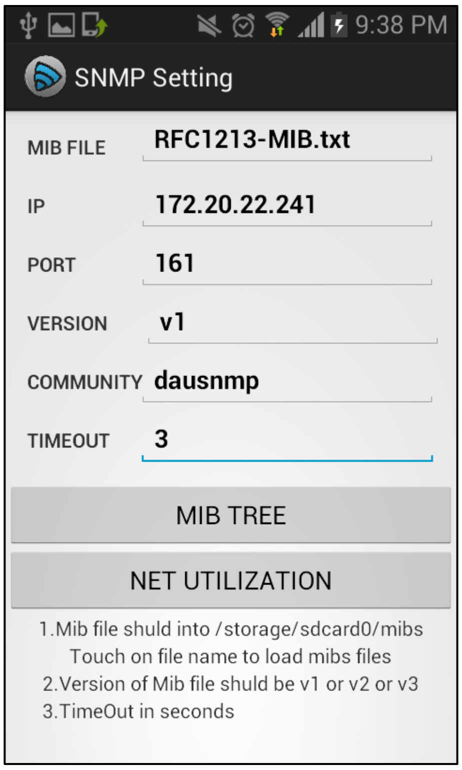


Figure 3.18. Sequence Diagram of SnmpSetting activity

Furthermore, the SnmpSetting activity can call SnmpTreeMake activity to build tree of MIB file by clicking on (MIB tree) button as shown in Figure 3.21.

Also, the SnmpSetting activity can call NetworkUtilization activity to calculate output utilization of wireless interface of the access point as shown in Figure 3.22.

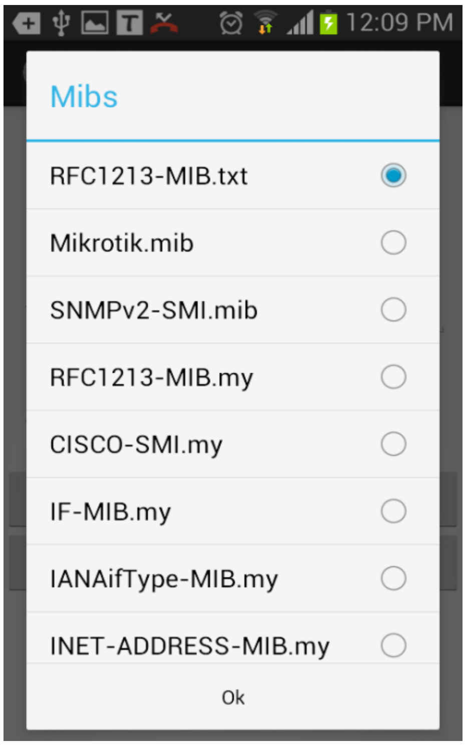

9:38 PM

SNMP Setting

MIB FILE	RFC1213-MIB.txt
IP	172.20.22.241
PORT	161
VERSION	v1
COMMUNITY	dausnmp
TIMEOUT	3

1.Mib file shuld into /storage/sdcard0/mibs
Touch on file name to load mibs files
2.Version of Mib file shuld be v1 or v2 or v3
3.TimeOut in seconds

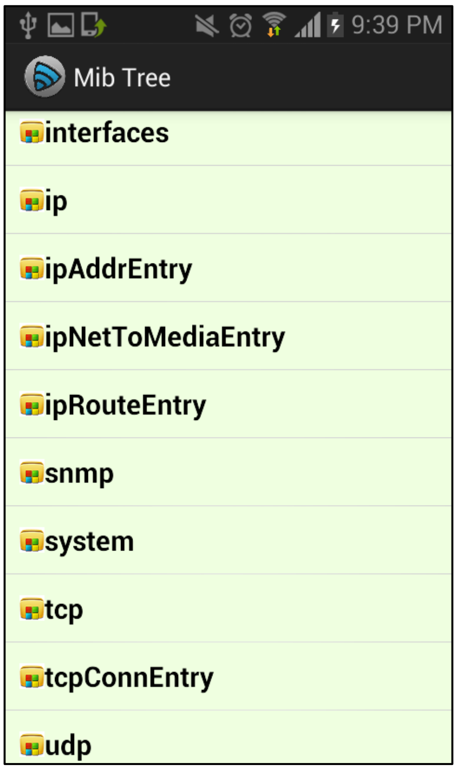
Figure 3.19. SnmpSetting activity


12:09 PM

Mibs

- RFC1213-MIB.txt ●
- Mikrotik.mib ○
- SNMPv2-SMI.mib ○
- RFC1213-MIB.my ○
- CISCO-SMI.my ○
- IF-MIB.my ○
- IANAifType-MIB.my ○
- INET-ADDRESS-MIB.my ○

Figure 3. 20. Dialog of MIB files


9:39 PM

Mib Tree

interfaces

ip

ipAddrEntry

ipNetToMediaEntry

ipRouteEntry

snmp

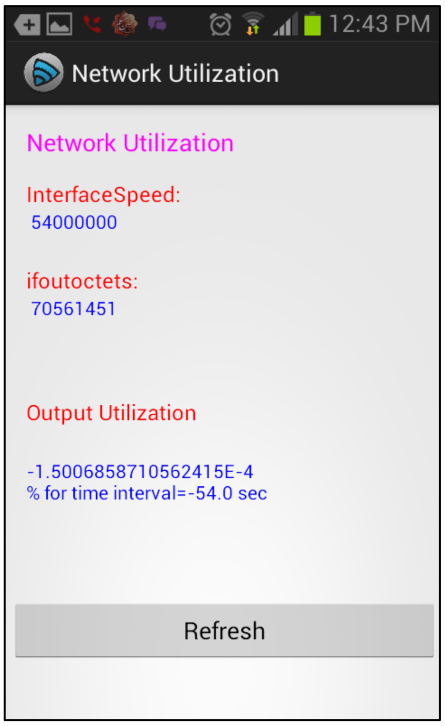
system

tcp

tcpConnEntry

udp

Figure 3. 21. SnmpTreeMake activity


12:43 PM

Network Utilization

Network Utilization

InterfaceSpeed:
54000000

ifoutoctets:
70561451

Output Utilization

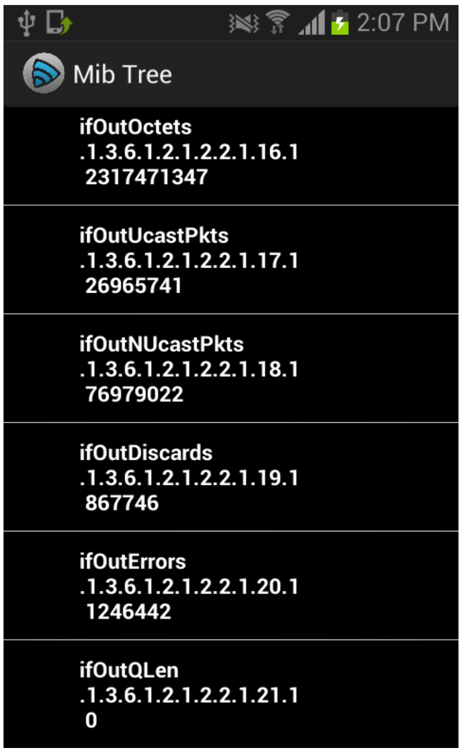
-1.5006858710562415E-4
% for time interval=-54.0 sec

Refresh

Figure 3.22. NetworkUtilization activity

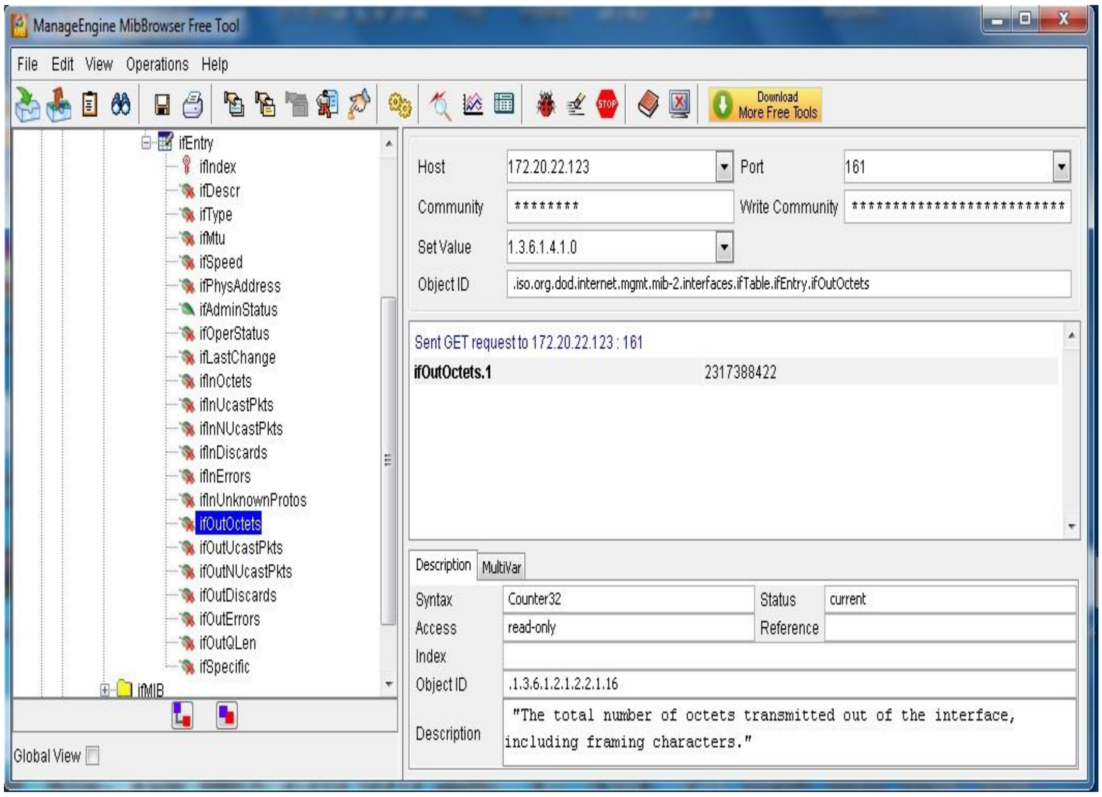
3.5 Comparison and Verification

We conducted an experiment to verify the implemented project and run EMUWiFiManager on Android device version 4.1.2 with MibBrowser desktop application on Windows 7. We chose the CMPE Library and connected to AP type Cisco AP C1240 with SSID CMPE LIBRARY WIFI. We obtained similar results in two applications as shown in Figure 3.23 and Figure 3.24.



```
ifOutOctets
.1.3.6.1.2.1.2.2.1.16.1
 2317471347

ifOutUcastPkts
.1.3.6.1.2.1.2.2.1.17.1
 26965741

ifOutNUcastPkts
.1.3.6.1.2.1.2.2.1.18.1
 76979022

ifOutDiscards
.1.3.6.1.2.1.2.2.1.19.1
 867746

ifOutErrors
.1.3.6.1.2.1.2.2.1.20.1
 1246442

ifOutQLen
.1.3.6.1.2.1.2.2.1.21.1
 0
```

Figure 3.23. Mib Tree on EMUWiFiManager Android Application



Host	172.20.22.123	Port	161
Community	********	Write Community	**************************
Set Value	1.3.6.1.4.1.0		
Object ID	.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOutOctets		

```
Sent GET request to 172.20.22.123 : 161
ifOutOctets.1                    2317388422
```

Syntax	Counter32	Status	current
Access	read-only	Reference	
Index			
Object ID	.1.3.6.1.2.1.2.2.1.16		
Description	"The total number of octets transmitted out of the interface, including framing characters."		

Figure 3.24. Mib Tree on MibBroser Windos 7 Application

3.6 Application Testing Framework

The Android application testing framework, an integral part of the development environment, gives an architecture and powerful tools that test every aspect of application.

Automated testing of Android application is important because of the large variety of available devices. As it is not practical to test Android application on all possible device configurations, it is common practice to run Android test on typical device configurations.

Having a practical test coverage for Android application helps us to enhance and maintain the Android application.

The user interface application tester Monkey [26], usually called "monkey", is a program that runs on emulator or device and generatesis pseudo-random streams as a command-line tool of keystrokes, touches, and gestures to a device. It returns a report back with errors and is run by Debug Bridge (adb) tool.

The following monkey testing sends 500 random events to the application with the allclasess.emuwifimanager package as shown in Figure 3.25.The final testing report is included in Appendix G.

```
adb shell monkey -p allclasess.emuwifimanager -v 500
```

where

adb – tool used to send command from a desktop or laptop computer to emulator or Android device.

shell - translates commands to OS commands.

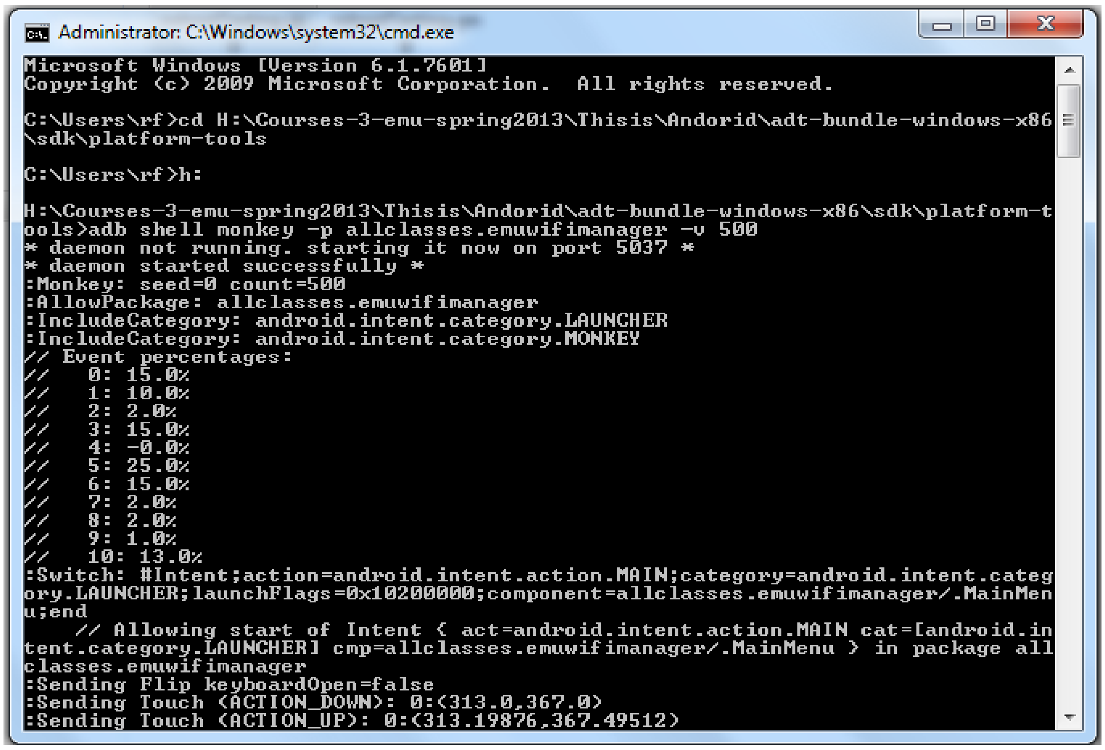
monkey - testing tool.

-p -as allowed-package-name.

v - stands for increment the verbosity level.

500- random number of event to be sent for testing application.

The event percentages in this random testing is illustrated in Table 3.1.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\rf>cd H:\Courses-3-emu-spring2013\Thisis\Andorid\adt-bundle-windows-x86
\sdk\platform-tools

C:\Users\rf>h:

H:\Courses-3-emu-spring2013\Thisis\Andorid\adt-bundle-windows-x86\sdk\platform-t
ools>adb shell monkey -p allclasses.emuwifimanager -v 500
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
:Monkey: seed=0 count=500
:AllowPackage: allclasses.emuwifimanager
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
//   0: 15.0%
//   1: 10.0%
//   2: 2.0%
//   3: 15.0%
//   4: -0.0%
//   5: 25.0%
//   6: 15.0%
//   7: 2.0%
//   8: 2.0%
//   9: 1.0%
//   10: 13.0%
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.categ
ory.LAUNCHER;launchFlags=0x10200000;component=allclasses.emuwifimanager/.MainMen
u;end
    // Allowing start of Intent { act=android.intent.action.MAIN cat=[android.in
tent.category.LAUNCHER] cmp=allclasses.emuwifimanager/.MainMenu } in package all
classes.emuwifimanager
:Sending Flip keyboardOpen=false
:Sending Touch (ACTION_DOWN): 0:(313.0,367.0)
:Sending Touch (ACTION_UP): 0:(313.19876,367.49512)
```

Figure 3.25. Monkey Testing Results

Table 3.2. The Event percentages Monkey Testing

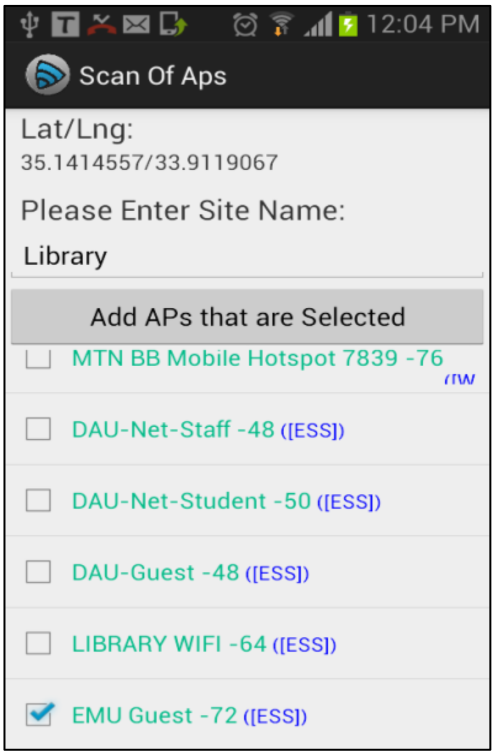
Event code	Event name	Percentage%
0	touch	15
1	motion	10
2	Trackball	2
3	Sys keys	15
4	navigation	0
5	Major nav	25
6	App switch	15
7	flip	2
8	Any event	2
9	Any event	1
10	Any event	13

As a result, the monkey testing shows no errors in EMUWiFiManager.

Chapter 4

A CASE STUDY WITH EMUWIFIMANAGER

We conducted an experiment to test the performance of the implemented project and run EMUWiFiManager on Android device version 4.1.2. We chose one of the EMU library's AP type Cisco AP 1200 with ssid EMU Guest (Figure 4.1) and setup SNMP with parameters as shown in Figure 4.2 and then MIB tree in Figure 4.3.


12:04 PM

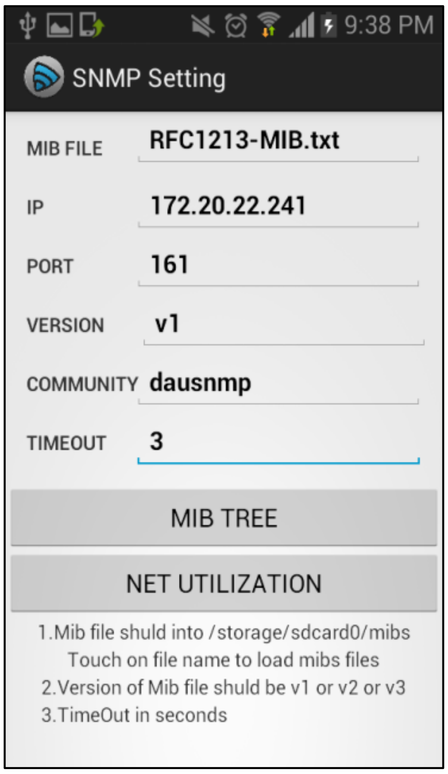
Scan Of Aps

Lat/Lng:
35.1414557/33.9119067

Please Enter Site Name:
Library

- [ ] MTN BB Mobile Hotspot 7839 -76 ([W
- [ ] DAU-Net-Staff -48 ([ESS])
- [ ] DAU-Net-Student -50 ([ESS])
- [ ] DAU-Guest -48 ([ESS])
- [ ] LIBRARY WIFI -64 ([ESS])
- [x] EMU Guest -72 ([ESS])

Figure 4.1. Connect to AP



9:38 PM

SNMP Setting

MIB FILE RFC1213-MIB.txt

IP 172.20.22.241

PORT 161

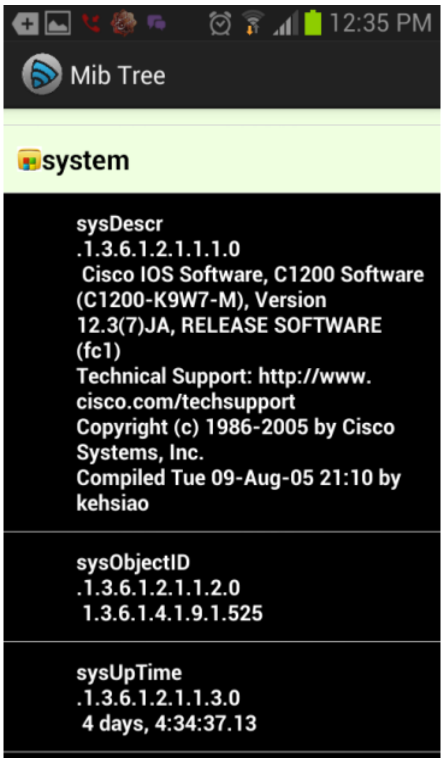
VERSION v1

COMMUNITY dausnmp

TIMEOUT 3

1.Mib file shuld into /storage/sdcard0/mibs
Touch on file name to load mibs files
2.Version of Mib file shuld be v1 or v2 or v3
3.TimeOut in seconds

Figure 4.2. Setup SNMP



12:35 PM

Mib Tree

system

```
sysDescr
.1.3.6.1.2.1.1.1.0
 Cisco IOS Software, C1200 Software
(C1200-K9W7-M), Version
12.3(7)JA, RELEASE SOFTWARE
(fc1)
Technical Support: http://www.
cisco.com/techsupport
Copyright (c) 1986-2005 by Cisco
Systems, Inc.
Compiled Tue 09-Aug-05 21:10 by
kehsiao

sysObjectID
.1.3.6.1.2.1.1.2.0
 1.3.6.1.4.1.9.1.525

sysUpTime
.1.3.6.1.2.1.1.3.0
 4 days, 4:34:37.13
```

Figure 4.3. Mib tree

4.1 Experiment Results

Our experiments focused on how to calculate network utilization and throughput by using Simple Network Management Protocol (SNMP).

4.1.1 Network Utilization

Network utilization is the amount of traffic on the network compared to the peak amount that the network can support. This is generally specified as a percentage.

By using MIB tree and then ifEntry (wireless interface table of AP) and then read values of ifSpeed (bandwidth Mbps) and ifOutOctets (output bytes) as shown in Figure 4.4.

ifSpeed is the maximum speed of wireless interface in that AP (54Mbps).

ifOutOctets are counts of the number of bytes output by the interface.

Utilization is calculated by the formula:

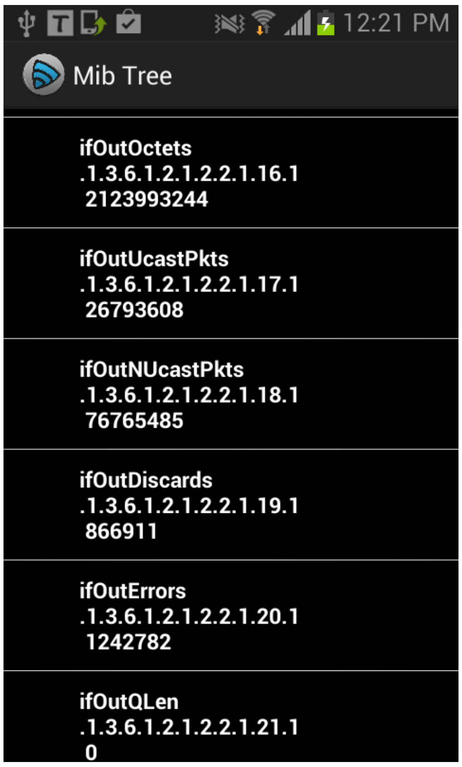
Network utilization % = (data bits * 100) / (bandwidth Mbps * interval in sec)

and then:

$$\text{Output utilization} = \frac{\Delta(\text{ifOutOctets}) * 8 * 100}{(\text{seconds in } \Delta) * \text{ifSpeed}}$$

where

Δ*ifOutOctets* is the difference between two reading of the ifOutOctets object, which represents outbound octets of network traffic.



```
ifOutOctets
.1.3.6.1.2.1.2.2.1.16.1
 2123993244

ifOutUcastPkts
.1.3.6.1.2.1.2.2.1.17.1
 26793608

ifOutNUcastPkts
.1.3.6.1.2.1.2.2.1.18.1
 76765485

ifOutDiscards
.1.3.6.1.2.1.2.2.1.19.1
 866911

ifOutErrors
.1.3.6.1.2.1.2.2.1.20.1
 1242782

ifOutQLen
.1.3.6.1.2.1.2.2.1.21.1
 0
```

Figure 4.4. Mib tree (ifEntry)

Here, we measured by reading from ifOutOctets (interface wireless output) OID every 5 min .

The first measurement was on 7/4/2014 from 11:00 AM to 11:40 AM and is given in Table 4.1 and presented in Figure 4.5.The measurement from 3:00 PM to 3:40 PM is given in Table 4.2 and presented in Figure 4.6.

The second measurement was on 8/4/2014 from 10:00 AM to 10:40 AM and is given in Table 4.3 and presented in Figure 4.7. The measurement from 2:00 PM to 2:40 PM is given in Table 4.4 and presented in Figure 4.8.

The third measurement was on 9/4/2014 from 10:00 AM to 10:40 AM and is given in Table 4.5 and presented in Figure 4.9.The measurement from 2:00 PM to 2:40 PM is given in Table 4.6 and presented in Figure 4.10.

The measurement outcomes show that the network utilization of this AP (Cisco AP 1200 with ssid EMU Guest) is low becuse not all students are using this AP in EMU library.

Table 4.1. ifOutOctets OID every 5 min date:7/4/2014 11:00AM

Date	**7/4/2014**		
time	**11:00 AM**		
first reading (Bytes)	**next reading (Bytes)**	**time**	Output utilization%
374632114	555732231	11:00	8.943215654
657338837	879348821	11:10	10.963456
997900877	1188500654	11:20	9.412334667
1188500654	1291788200	11:30	5.100619556
1350100321	1467991089	11:40	5.821766321

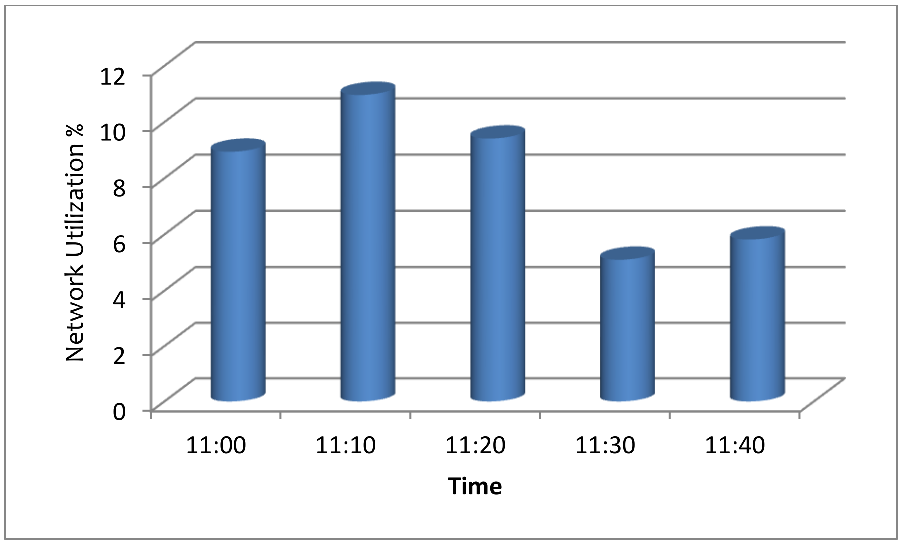


Figure 4.5. Network utilization at date 7/4/2014 11:00AM

Table 4.2. ifOutOctets OID every 5 min date:7/4/2014 3:00PM

Date	7/4/2014		
time	3:00 PM		
first reading (Bytes)	next reading (Bytes)	time	Output utilization%
4280076999	4380099925	3:00	4.939403753
4390099122	4498765430	3:10	5.366237432
4607314111	4707431749	3:20	4.944080889
4807463155	4899500999	3:30	4.545078716
4999555405	5087908321	3:40	4.363106963

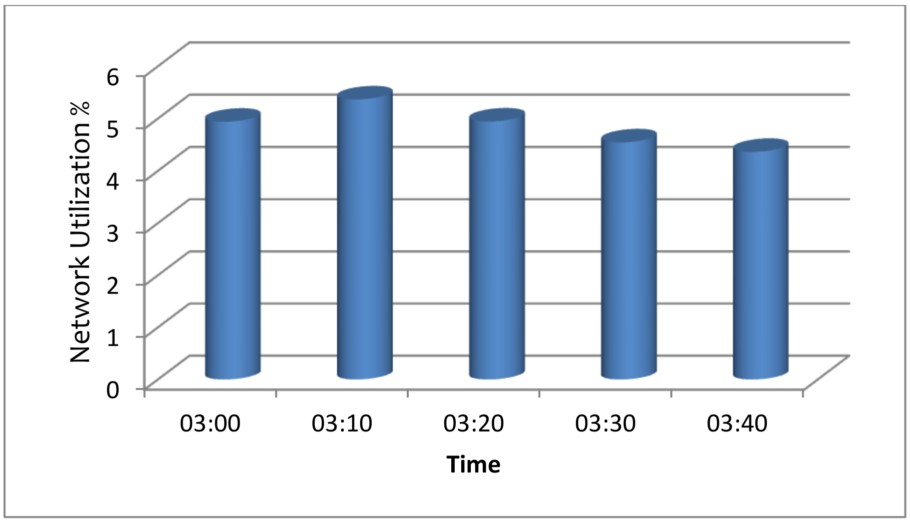


Figure 4.6. Network utilization at date 7/4/2014 3:00PM

Table 4.3. ifOutOctets OID every 5 min date:8/4/2014 0:00AM

Date	8/4/2014		
time	10:00 AM		
first reading (Bytes)	next reading (Bytes)	time	Output utilization%
7270481898	7399481221	10:00	6.370336938
7528480544	7650870534	10:10	6.043950123
7788290567	7911300435	10:20	6.074561383
8043690425	8190300666	10:30	7.240011901
8400780786	8577911899	10:40	8.747215457

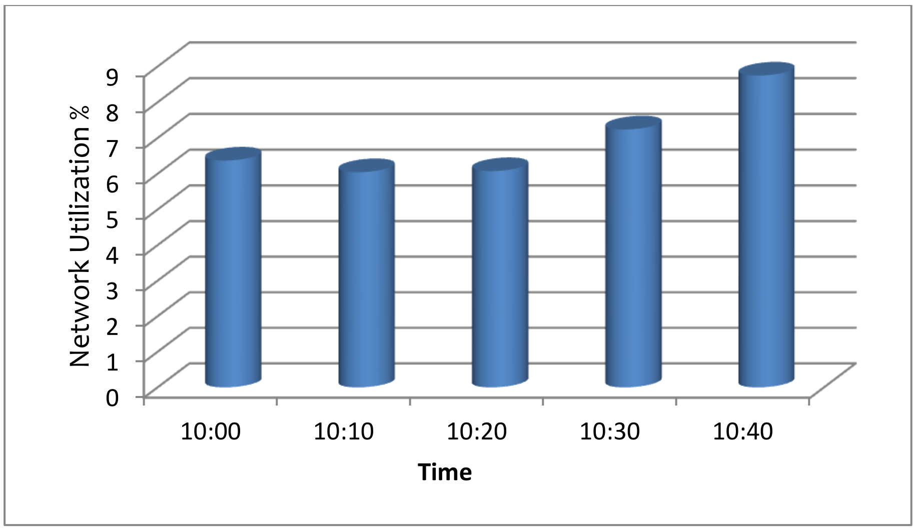


Figure 4.7. Network utilization at date 8/4/2014 10:00AM

Table 4.4. ifOutOctets OID every 5 min date:8/4/2014 2:00PM

date	8/4/2014		
time	2:00 PM		
first reading (Bytes)	next reading (Bytes)	time	Output utilization%
11225711699	11366644345	2:00	6.95963684
11607576991	11740511000	2:10	6.56464242
11893445009	11991006753	2:20	4.817863901
12088568497	12110787191	2:30	1.097219457
12133005885	12156321117	2:40	1.151369481

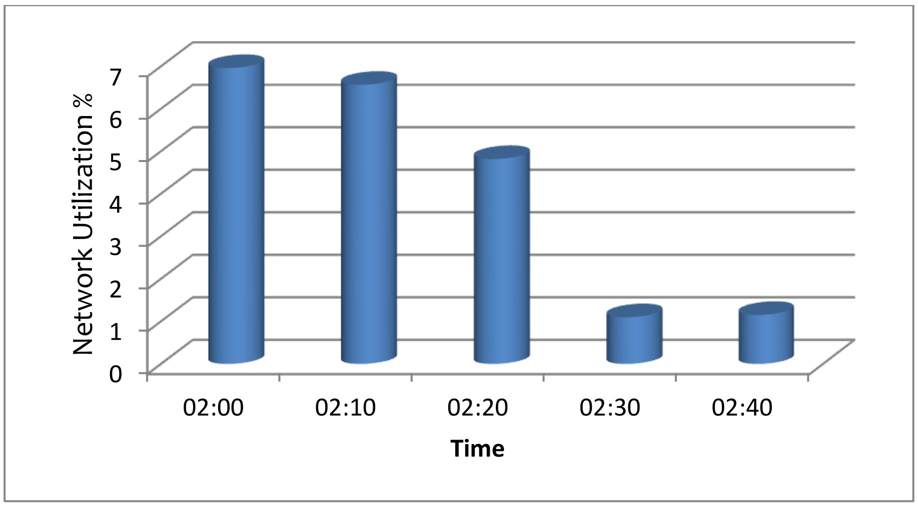


Figure 4.8. Network utilization at date 8/4/2014 2:00PM

Table 4.5. ifOutOctets OID every 5 min date:9/4/2014 10:00AM

Date	9/4/2014		
time	10:00 AM		
first reading (Bytes)	next reading (Bytes)	time	Output utilization%
189150558	311240988	10:00	6.029157037
433355418	571455399	10:10	6.819752148
709599380	859423001	10:20	7.398697333
1009456785	1210588632	10:30	9.932436889
1500720479	1699427917	10:40	9.812712988

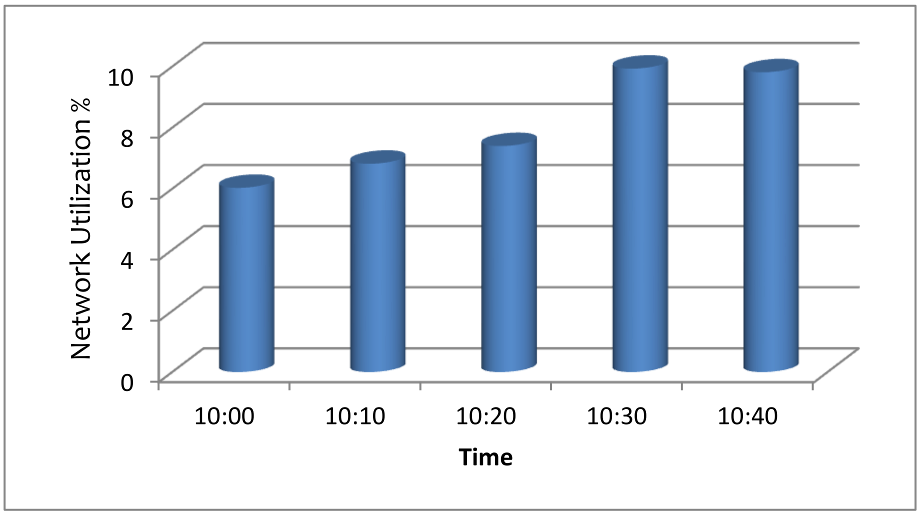


Figure 4.9. Network utilization at date 9/4/2014 10:00AM

Table 4.6. ifOutOctets OID every 5 min date:9/4/2014 2:00PM

date	9/4/2014		
time	2:00 PM		
first reading (Bytes)	next reading (Bytes)	time	Output utilization%
4530832077	4644391102	2:00	5.607853086
4757950127	4855876372	2:10	4.835863951
4953802617	5067711300	2:20	5.625120148
5181719983	5290726010	2:30	5.383013679
5400732037	5511901088	2:40	5.489829679

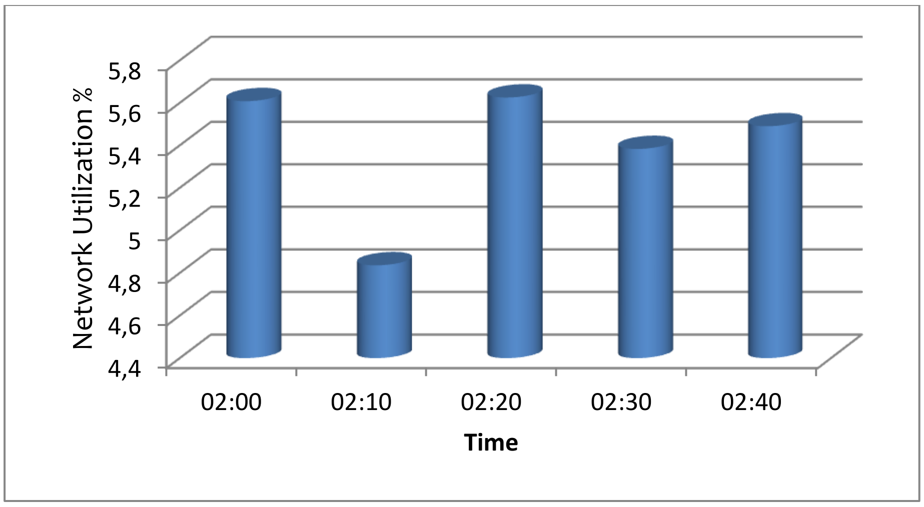
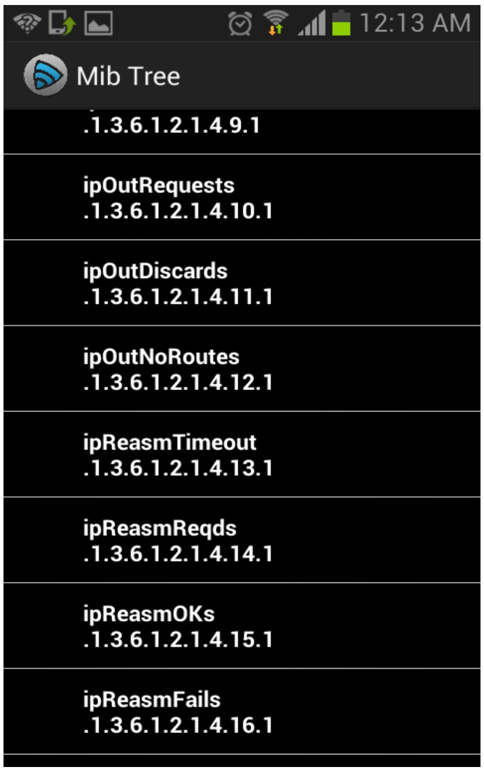


Figure 4.10. Output network utilization at date 9/4/2014 and time 2:00PM

4.1.2 Throughput

Network throughput is the rate of successful message delivery over a communication channel. The throughput is usually measured in bits per time unit, and sometimes in data packets per time unit or data packets per time slot.

From the MIB, we read values of ipOutRequests OID (packets count) (Figure 4.11).



12:13 AM

Mib Tree

.1.3.6.1.2.1.4.9.1

ipOutRequests
.1.3.6.1.2.1.4.10.1

ipOutDiscards
.1.3.6.1.2.1.4.11.1

ipOutNoRoutes
.1.3.6.1.2.1.4.12.1

ipReasmTimeout
.1.3.6.1.2.1.4.13.1

ipReasmReqds
.1.3.6.1.2.1.4.14.1

ipReasmOKs
.1.3.6.1.2.1.4.15.1

ipReasmFails
.1.3.6.1.2.1.4.16.1

Figure 4.11. Mib tree (IP)

The measurements were taken on 20/6/2014 from 10:00 AM to 10:40 AM and are given in Table 4.7 and presented in Figure 4.12.The measurements from 2:00 PM to 2:40 PM are given in Table 4.8 and presented in Figure 4.13.

Table 4.7. ipOutRequests OID every 5 min date:20/6/2014 10:00AM

Date	20/6/2014	IP	
time	10:00 AM		
packets	packets	time	IP throughput (packets/min)
308478	309586	10:00	277
309952	310990	10:10	259
311792	312712	10:20	230
312942	313992	10:30	262
314400	315358	10:40	239

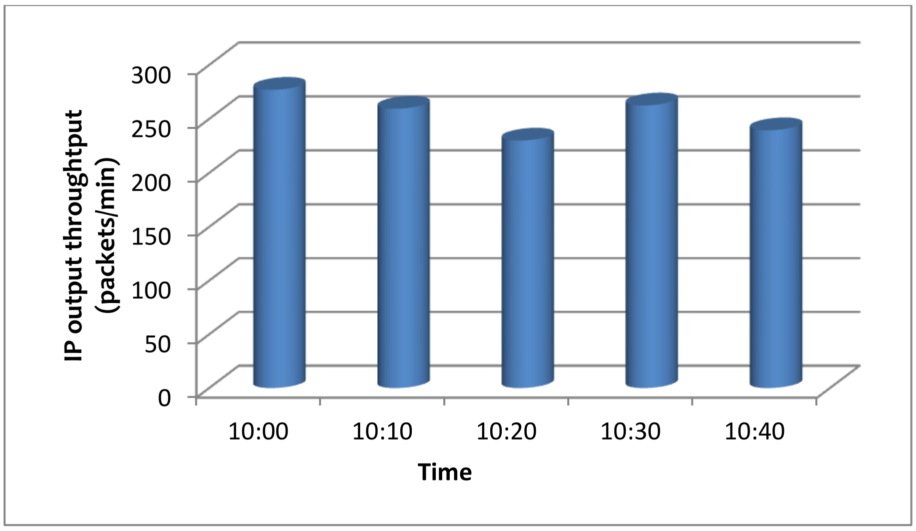


Figure 4.12. ipOutRequests OID at date 20/6/2014 and time 10:00AM

Table 4.8. ipOutRequests OID every 5 min date:20/6/2014 2:00PM

date	20/6/2014		
time	2:00 PM		
packets	packets	time	IP throughput (packets/min)
308478	309586	2:00	221
309952	310990	2:10	207
311792	312712	2:20	184
312942	313792	2:30	170
314400	315358	2:40	191

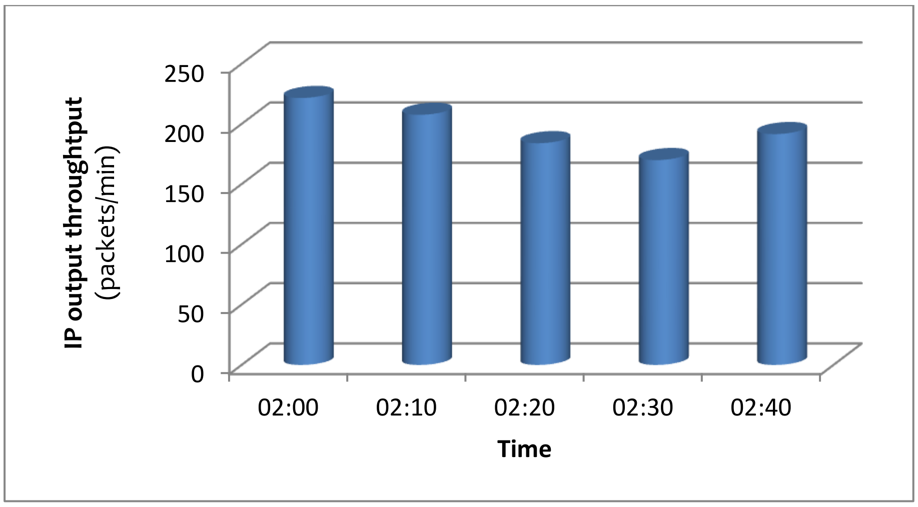
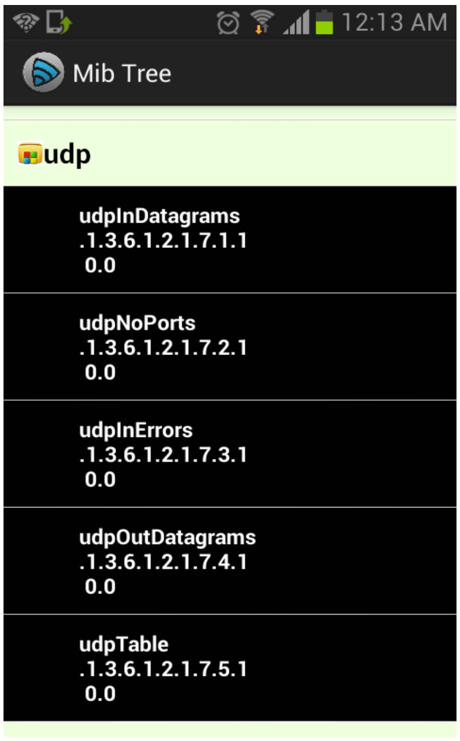


Figure 4.13 ipOutRequests OID at date 20/6/2014 and time 2:00AM

By using MIB, we read the values of udpOutDatagrams OID (datagram count) (Figure 4.14).


12:13 AM

Mib Tree

udp

udpInDatagrams
.1.3.6.1.2.1.7.1.1
0.0

udpNoPorts
.1.3.6.1.2.1.7.2.1
0.0

udpInErrors
.1.3.6.1.2.1.7.3.1
0.0

udpOutDatagrams
.1.3.6.1.2.1.7.4.1
0.0

udpTable
.1.3.6.1.2.1.7.5.1
0.0

Figure 4.14. Mib tree (udp)

The measurements were taken on 20/6/2014 from 10:00 AM to 10:40 AM and are given in Table 4.9 and presented in Figure 4.15. The measurements from 2:00 PM to 2:40 PM are given in Table 4.10 and presented in Figure 4.16.

Table 4.9. udpOutDatagrams OID every 5 min date:20/6/2014 10:00AM

Date	20/6/2014		
time	10:00 AM		
datagrams	datagrams	time	UDP Output throughput (datagram/min)
307976	308972	10:00	199
309838	310812	10:10	194
313686	314400	10:20	142
312744	313470	10:30	145
314264	314946	10:40	136

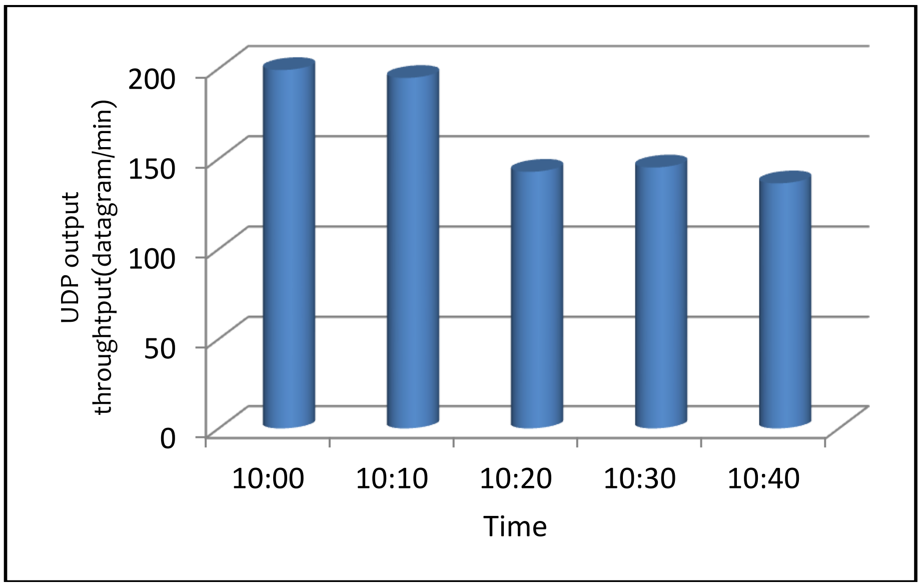


Figure 4.15. udpOutDatagrams OID at date 20/6/2014 and time 10:00AM

Table 4.10. udpOutDatagrams OID every 5 min date:20/6/2014 2:00PM

Date	20/6/2014		
time	2:00 PM		
datagrams	datagrams	time	UDP Output throughput (datagrams/min)
200184	200831	2:00	129
232378	233109	2:10	146
311686	312114	2:20	85
187646	188081	2:30	87
235698	236209	2:40	102

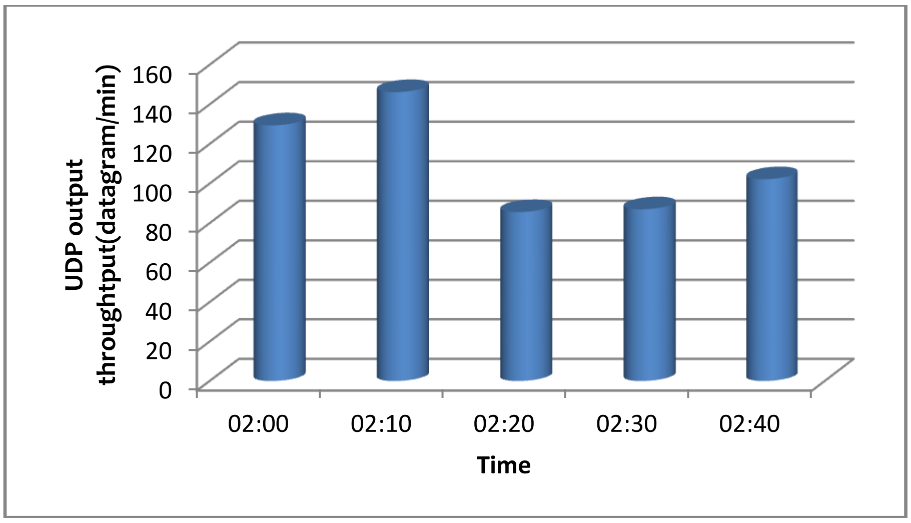
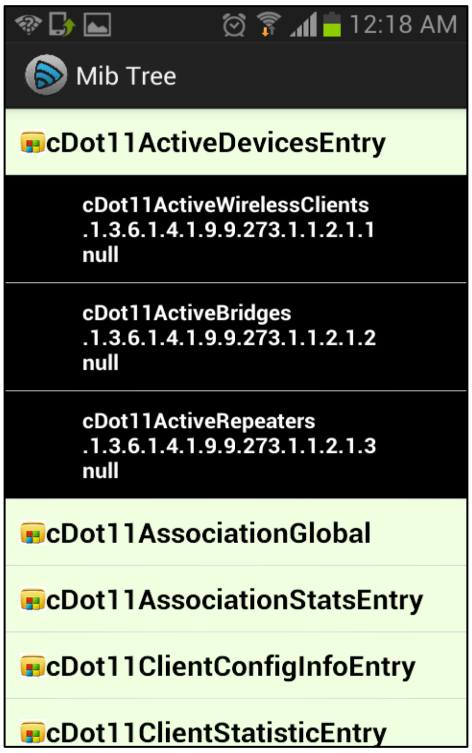


Figure 4.16. udpOutDatagrams OID at date 20/6/2014 and time 2:00PM

4.1.3 Number of Active Wireless clients

By using MIB tree and then cDot11ActiveDevicesEntry, we read values of cDot11ActiveWirelessClients OID (devices count) (Figure 4.17).



cDot11ActiveDevicesEntry

cDot11ActiveWirelessClients
.1.3.6.1.4.1.9.9.273.1.1.2.1.1
null

cDot11ActiveBridges
.1.3.6.1.4.1.9.9.273.1.1.2.1.2
null

cDot11ActiveRepeaters
.1.3.6.1.4.1.9.9.273.1.1.2.1.3
null

cDot11AssociationGlobal

cDot11AssociationStatsEntry

cDot11ClientConfigInfoEntry

cDot11ClientStatisticEntry

Figure 4.17. Mib tree (cDot11ActiveDevicesEntry)

The measurements were taken on 20/6/2014 from 10:00 AM to 10:40 AM and are given in Table 4.11. The measurement from 2:00 PM to 2:40 PM is given in Table 4.12.

Table 4.11. cDot11ActiveWirelessClients every 5 min date:20/6/2014 10:00AM

Date	20/6/2014	
time	10:00 AM	
Active Devices (beginning of interval)	Active Devices(end of interval)	time
12	10	10:00
6	8	10:10
8	8	10:20
10	9	10:30
10	10	10:40

Table 4.12. cDot11ActiveWirelessClients every 5 min date:20/6/2014 2:00PM

date	20/6/2014	
time	2:00 PM	
Active Devices(beginning of interval)	Active Devices(end of interval)	time
6	5	2:00
6	5	2:10
7	6	2:20
9	9	2:30
8	8	2:40

4.2 Discussion of Results

Prior to the installation of any wireless devices, administrators can review the usage requirements for the area in question to determine the optimum number of wireless access points needed to efficiently support all users in the area simultaneously. The EMUWiFiManager application used to site survey EMU library area revealed useful data for planning of WiFi access points and requirements, if any, for expansion.

Utilization measurements conducted on different times on different days showed us that the access point under consideration was not heavily utilized: The utilization varied between 1% and 11%. These results may imply that the number of access points installed in the library for the number of users is currently sufficient.

To verify the utilization measurements, the number of IP packets sent out by the device was also recorded and it was found that the number of packets per minute was indeed low. Furthermore, roughly half or more of these packets were UDP packets. Most of these UDP packets may be associated with DNS queries. To have a more accurate picture of composition of traffic, more sophisticated network monitoring tools may be necessary.

The number of wireless clients attached to the access point varied between 5 and 12. This may indicate that the number of people using WiFi in the library is not that high. To have a more accurate picture, permission to monitor all access points must be obtained and, if possible, users may be surveyed to find out what kind of Internet connection they prefer when they connect to the Internet (WiFi or 3G).

Chapter 5

CONCLUSION

A main aspect of ubiquitous computing is context-aware applications where the applications collect information about the environment that the user is in and use this information to achieve their goals or improve performance.

The solution presented in this project is useful enough to assist network managers in managing their wireless networks. However, it may appear cumbersome for many network managers to store information about their wireless networks manually into the application.

At the time of writing, there was no standard mobile application to visualize and manage all Wi-Fi access points at EMU that gives the network managers the information of both the locations of the Wi-Fi access points, as well as its congestion profiles. The viable improvement would be get all information and locations of Wi-Fi access points by entering a few of parameters and touching on screens of Android mobile application (EMUWiFiManager).

The most important tools in the construction of this project is to show the access points on Google Maps and then get access point information using the SNMP protocol. Hence, we used many of API libraries that can runs on Android mobile system.

To illustrate the benefit of the use of this project, we conducted a number of measurements for network utilization and throughput near the EMU library. Wireless access service can be provided on the basis of anticipated utilization data gathered during initial site surveys conducted by administrators. As the number of users increases, site surveys can be repeated to have an up-to-date information about the network.

For further work, we recommend more focus on SNMP protocol to manage all devices of the network. In addition, further work is needed not only retrieve data but to change parameters for these devices according to its environment.

This application code can be improved in three aspects:

1. Improving the efficiency of threading
2. Optimizing device battery life
3. Minimizing memory use in the UI

It is also desible to port the application to other mobile platforms such as iOS and Windows Phone. Doing so will provide access to this application to a wider range of users.

Finally, possibility of adding extra features to this application should be investigated. In particular, signal strength measurement facility can be integrated to the application.

REFERENCES

[1] O. Olusegun and C. E. Tan, "TCP throughput efficiency enhancement in IEEE 802.11n network," in *IEEE International Conference on Information Technology in Asia*, Malaysia, 2013.

[2] T. S. Rappaport, Wireless Communications: Principles and Practice (2nd Edition), New Jersey: Prentice Hall, 2002, pp. 1-327.

[3] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine,* vol. 51, no. 2, pp. 114-119, 2013.

[4] Z. Xu, Z. Chen and H. Nie, "Handheld Computers: Smartphone-Centric Wireless Applications," *IEEE Microwave Magazine,* vol. 15, no. 2, pp. 36- 44, 2014.

[5] S. S. A. C. S. Nimodia, "A Survey on Network Monitoring and Administration Using Email and Android Phone," *International Journal of Emerging Technology and Advanced Engineering,* vol. 3, no. 4, pp. 83-87, 2013.

[6] Y. Yemini, A. Konstantinou and D. Florissi, "NESTOR: an architecture for network self-management and organization," *IEEE Journal on Selected Areas in Communications,* vol. 18, no. 5, pp. 758- 766, 2000.

[7] W. Enck, T. Moyer, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, Y.-W. E. Sung, S. Rao and W. Aiello, "Configuration management at massive scale: system design and experience," *IEEE Journal on Selected Areas in Communications,* vol. 27, no. 3, pp. 323- 335, 2009.

[8] H. Li, "Wireless LAN Network Management System," in *2004 IEE ,Industrial Electronics*, San Jose, 2004.

[9] A. Liu and A. Khakpour, "Quantifying and Verifying Reachability for Access Controlled Networks," *IEEE TRANSACTIONS ON NETWORKING,* vol. 21, no. 2, pp. 551-565, 2013.

[10] C.-C. Teng and R. Helps, "Mobile Application Development: Essential New Directions for IT," in *IEEE,Seventh International Conference on Information Technology*, Utah, 2010.

[11] Z. D. R. C. Xianhua Shu, "Research on Mobile Location Service Design Based on Android," in *IEEE,Wireless Communications, Networking and Mobile Computing*, Dalian, 2009.

[12] C. C. A. R. S. W. Frank Ableson, Unlocking Android A Developer's Guide, New York: Manning, 2009, pp. 1-24.

[13] tutorialspoint, "Android Architecture," 9 3 2014. [Online]. Available: http://www.tutorialspoint.com/android/android_architecture.htm. [Accessed 9 3

2014].

[14] E. Burnette, Hello, Android,Introducing Google's Mobile Development Platform, 3rd Edition, Texas: The Pragmatic Programmers, 2010.

[15] M. Z. Robert Green, Beginning Android Games, Portland: APress, 2012, p. 714.

[16] S. Conder and L. Darcey, Android Wireless Application Development 2nd, Michigan: Addison-Wesley Professional, 2010.

[17] A. Developers, "Get the Android SDK," 5 3 2013. [Online]. Available: http://developer.android.com/sdk/index.htm. [Accessed 10 3 2014].

[18] "Eclipsepedia," 16 1 2014. [Online]. Available: http://wiki.eclipse.org/Main_Page. [Accessed 14 3 2014].

[19] ORACLE, "How do I get Java for Mobile device," 27 1 2014. [Online]. Available: http://www.java.com/en/download/faq/java_mobile.xml. [Accessed 14 3 2014].

[20] W3C, "Extensible Markup Language (XML)," 29 10 2013. [Online]. Available: http://www.w3.org/XML/. [Accessed 14 3 2014].

[21] Hwaci, "About SQLite," 13 3 2013. [Online]. Available:

http://www.sqlite.org/about.html. [Accessed 14 3 2014].

[22] "Android Wi-Fi," 14 3 2014. [Online]. Available: http://developer.android.com/reference/android/net/wifi/package-summary.html. [Accessed 17 3 2014].

[23] G. Svennerberg, Beginning Google Maps API 3, New York: APress, 2010, p. 328.

[24] D. R. S. K. J. Mauro, Essential SNMP, Sebastopol: O'Reilly, 2005.

[25] A. Frisch, "Essential System Administration, Third Edition," in *Essential System Administration, Third Edition*, California, O’Reilly, 2002, p. 1178.

[26] A. Developers, «UI/Application Exerciser Monkey,» 8 2 2014. [Çevrimiçi]. Available: http://developer.android.com/tools/help/monkey.html. [%1 tarihinde erişilmiştir25 4 2014].

APPENDICES

Appendix A: MainMenu Class

```
package allclasses.emuwifimanager;
import allclasses.emuwifimanager.R;
import android.app.Activity;//to Inheritance Activity
import android.content.Intent;//for call another class
import android.os.Bundle;//os bundle
import android.view.View;
 public class MainMenu extends Activity {
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.mainm);
            }
            public void viewemumapclk(View view ) {
                          Intent  intent =new Intent( this,allclasses.emuwifimanager.EmuMap.class);
                         startActivity(intent);

            }

            public void viewlistsitesclk(View view ) {
            Intent intent =new ntent(MainMenu.this,allclasses.emuwifimanager.SiteSearching.class);
                        startActivity(intent);
            }
            public void scanapsclk(View view ) {
                        Intent  intent =new
                    Intent(MainMenu.this,allclasses.emuwifimanager.ScanOfAps.class);
                        startActivity(intent);
            }

            public void snmpsettingclk(View view ) {
                        Intent  intent =new
                     Intent(MainMenu.this,allclasses.emuwifimanager.SnmpSetting.class);
                        startActivity(intent);

            }
            public void clk2(View v )
            {
            this.finish();
            }
}
```

Appendix B: ScanOfAps Class

```
package allclasses.emuwifimanager;
import java.text.SimpleDateFormat;
import java.util.ArrayList;// array list of object
import java.util.Date;// date and time
import java.util.List;// array list of object
import allclasses.emuwifimanager.R;
import android.annotation.SuppressLint;
import android.app.Activity; //to Inheritance Activity
import android.app.AlertDialog;//for deleting
import android.app.Dialog; //to enter IP
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent; ;//for call another class
import android.content.IntentFilter;
import android.location.Criteria;
import android.location.Location;//lat/lng
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.wifi.ScanResult;//wi fi APs
import android.net.wifi.WifiManager;//APs infromation
import android.os.Bundle; //os bundle
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;//add APs
import android.widget.CheckBox;//select APs
import android.widget.EditText;//site name
import android.widget.ListView;//views APs
import android.widget.TextView;//Lat/Lng
import android.widget.Toast;//message box
public class ScanOfAps extends Activity implements LocationListener
{

            //ListViewCheckboxesActivity
            CustomAdapter dataAdapterScanAPs = null;
   WifiManager mainWifi;
   WifiReceiver receiverWifi;
   String[] AP_Name;
   String[] AP_Level;
   String[] AP_Status;
   String[] AP_Macaddress;

   List<ScanResult> wifiList;
   ArrayList<Apsinfo> apsinfoArrayList;
   ArrayList<String> iplist;
   Button btnAddAPs;
   ApsApplication apsApplication;

   Double lat,lng;
   EditText edittextSiteName;
   TextView  txtlatlng;
   ListView listViewOfScanAPs;
   Apsinfo apsInfo;
   private LocationManager locationManager;
   private String locationProvider;
   SimpleDateFormat formatter;;
   private static final int DLG_FlAG = 0;
   private static final int TEXT_ID_DLG = 0;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.mainlist);
     iplist = new  ArrayList<String>();

      apsApplication=new ApsApplication();

            apsApplication.apsDataBase=openOrCreateDatabase("myDB1",MODE_PRIVATE , null);
     edittextSiteName= (EditText) findViewById(R.id.site1);
```

```
          formatter= new SimpleDateFormat("dd/MM/yyyy");
        Toast.makeText(getApplicationContext(),"Please    waite    will    scaning    WiFi    APs...",
Toast.LENGTH_LONG).show();
        mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

        if (mainWifi.isWifiEnabled() == false)
        {
            Toast.makeText(getApplicationContext(),    "wifi    is    disabled..making    it    enabled",
Toast.LENGTH_LONG).show();
            mainWifi.setWifiEnabled(true);
        }

        receiverWifi = new WifiReceiver();
        registerReceiver(receiverWifi,new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        mainWifi.startScan();
        initializeGPSData();

        checkButtonClick();

        txtlatlng.setOnTouchListener ( new View.OnTouchListener() {

                                             @Override
                                             public boolean onTouch(View v, MotionEvent event) {
                                                          int eventaction = event.getAction();

                                                 switch (eventaction) {

                                                     case MotionEvent.ACTION_UP:

                                             //Intent  intent =new Intent(
                                             //ScanOfAps.this,NewLocationPointRetrieve.class);
                                                                //          startActivity(intent);
                                                          break;
                                                 }


                                                          return true;
                                             }


                                 });



      }

           private void displayListView()
     {

          ArrayList<Apsinfo> stateList = new ArrayList<Apsinfo>();
         for (int i=0;i<AP_Name.length-1;i++){
              Apsinfo          _states          =          new          Apsinfo(AP_Status[i],
AP_Name[i],AP_Macaddress[i],AP_Level[i],false);
               stateList.add(_states);
         }
```

```
            ////make  ArrayAdaptar from String Array
             dataAdapterScanAPs = new CustomAdapter(this,R.layout.listinfo, stateList);

            listViewOfScanAPs = (ListView) findViewById(R.id.listView1);
            ////make  adapter to ListView
            listViewOfScanAPs.setAdapter(dataAdapterScanAPs);

           }

private class CustomAdapter extends ArrayAdapter<Apsinfo>
{

private ArrayList<Apsinfo> apsInfoArray;

public CustomAdapter(Context context, int textViewResourceId,

ArrayList<Apsinfo> stateList)
{
  super(context, textViewResourceId, stateList);
  this.apsInfoArray = new ArrayList<Apsinfo>();
  this.apsInfoArray.addAll(stateList);
}

private class ViewHolderChkBoxSsidSNRWEP
{
TextView wepSecType;
CheckBox chkBoxssidSNR;

}

@SuppressLint("CutPasteId")
@Override
public View getView(int position, View convertView, ViewGroup parent)
{

        ViewHolderChkBoxSsidSNRWEP holder = null;

      if (convertView == null)
      {

         LayoutInflater vi =
(LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE);

         convertView = vi.inflate(R.layout.listinfo, null);

          holder = new ViewHolderChkBoxSsidSNRWEP();
        holder.wepSecType = (TextView) convertView.findViewById(R.id.code);
        holder.chkBoxssidSNR = (CheckBox) convertView.findViewById(R.id.checkBox1);

        convertView.setTag(holder);

               holder.chkBoxssidSNR.setOnClickListener( new View.OnClickListener()
               {

                    @SuppressWarnings("deprecation")
                                         public void onClick(View v)
                   {
                               btnAddAPs.setEnabled(true);

                   CheckBox cb = (CheckBox) v;
```

```
                    apsInfo = (Apsinfo) cb.getTag();
                   apsInfo.setSelected(cb.isChecked());


                   if (cb.isChecked()){ showDialog(DLG_FlAG);  }
                   Toast.makeText(getApplicationContext(),cb.getText() + " is " + cb.isChecked()  ,
                   Toast.LENGTH_LONG).show();


                 }
           });

     }
     else
     {
        holder = (ViewHolderChkBoxSsidSNRWEP) convertView.getTag();
     }

     Apsinfo apinfo = apsInfoArray.get(position);
     holder.wepSecType.setText(" (" + apinfo.getCode() + ")");//WEP sec
     holder.chkBoxssidSNR.setText(apinfo.getName()+" "+apinfo.getLevel());
     holder.chkBoxssidSNR.setChecked(apinfo.isSelected());
     holder.chkBoxssidSNR.setTag(apinfo);

     return convertView;
}

}

 private void checkButtonClick()
 {


        btnAddAPs = (Button) findViewById(R.id.findSelected);

       btnAddAPs.setOnClickListener(new OnClickListener()
       {

              @Override
              public void onClick(View v)
              {

                     StringBuffer textResponse = new StringBuffer();
                     textResponse.append("Add APs to Database that  were selected...\n");

                     ArrayList<Apsinfo> apsinfoAry = dataAdapterScanAPs.apsInfoArray;

                     for(int i=0;i<apsinfoAry.size();i++)
                     {
                         Apsinfo apsinfo = apsinfoAry.get(i);
                        if(apsinfo.isSelected())
                        {


                            apsApplication.apsDataBase.execSQL("INSERT    INTO    Accesspointss
VALUES(NULL,'"+apsinfo.getName()+"','"+
                            apsinfo.getMacaddress()+"',  '"+apsinfo.getCode()+"',  '"+lat.toString()+"',
'"+lng.toString()+"','"+
                             edittextSiteName.getText().toString()+"',            '"+formatter.format(new
Date()).toString()+"','"+apsinfo.getLevel()+"','"+apsinfo.ip4+"');");
```

```
                                textResponse.append("\n" +
                                          apsinfo.getName()+

                                ","+ apsinfo.getMacaddress()+

                                ","+apsinfo.getCode()+
                                 ","+lat.toString()+
                                 ","+lng.toString()+
                                 ","+edittextSiteName.getText().toString()+
                                ","+formatter.format(new Date()).toString()+
                                ","+apsinfo.getLevel()+","+ apsinfo.ip4);

                              }
                            }

                            Toast.makeText(getApplicationContext(),
                            textResponse, Toast.LENGTH_LONG).show();
                    }
            });
    }

    protected void onPause() {

        unregisterReceiver(receiverWifi);
        super.onPause();

        locationManager.removeUpdates(this);

    }

    protected void onResume() {

            registerReceiver(receiverWifi,new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        super.onResume();
    }
    class WifiReceiver extends BroadcastReceiver {
        @SuppressLint("UseValueOf")

                    public void onReceive(Context c, Intent intent)
        {


              wifiList = mainWifi.getScanResults();
             AP_Name= new String[wifiList.size()] ;
             AP_Status= new String[wifiList.size()] ;
             AP_Macaddress=new String[wifiList.size()] ;
             AP_Level= new String[wifiList.size()] ;
             apsinfoArrayList = new ArrayList<Apsinfo>();
             int i;
             //int j=0;
             for( i = 0; i < wifiList.size(); i++){
                     AP_Name[i]=wifiList.get(i).SSID;

                     AP_Level[i]=wifiList.get(i).level+"";
                     AP_Status[i]=wifiList.get(i).capabilities;
                     AP_Macaddress[i]=wifiList.get(i).BSSID;
                     //j++;
              }
             displayListView();
        }
    }


    @Override
    public void onLocationChanged(Location location) {

                showLocation(location);
                Toast.makeText(getApplicationContext(),"location                                   change",
Toast.LENGTH_LONG).show();

    }

    //@SuppressWarnings("static-access")
private void initializeGPSData() {

                txtlatlng = (TextView) findViewById(R.id.latlngv);
                 // gps = (TextView) findViewById(R.id.GPS);
                 locationManager                                                                          =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
                 //locationProvider = locationManager.GPS_PROVIDER;
                 locationProvider = locationManager.getBestProvider(new Criteria(), true);


                 if(locationProvider!=null)
                  {
                         //txtlatlng.setText("Upload image location: "+ locationProvider);
                         Toast.makeText(getApplicationContext(),"Upload     image     location:     "+
locationProvider, Toast.LENGTH_LONG).show();


                         Location location = locationManager.getLastKnownLocation(locationProvider);
                          lat = location.getLatitude();
                     lng = location.getLongitude();

                     txtlatlng.setText(""+lat+"/"+lng);

                  }
                 else
                 {
                     txtlatlng.setText("Upload image location: No provider Found" );
                         Toast.makeText(getApplicationContext(),"Upload   image   location:   No   provider
Found", Toast.LENGTH_LONG).show();
                 }



             }

    private void showLocation(Location location)
    {
        if(location==null)
                 txtlatlng.append("\nUnknown location");
        else
        {
            double lat = location.getLatitude();
            double lng = location.getLongitude();
```

```
          txtlatlng.setText(""+lat/+lng);

      }
  }

@Override
public void onProviderDisabled(String provider) {
                         txtlatlng.append("\nProvider Disabled: " +provider);
}


@Override
public void onProviderEnabled(String provider) {
             txtlatlng.append("\nProvider Enabled: " +provider);
}


@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
             txtlatlng.append("\nProvider Status Changed: " + provider+ ",status: "+
              status);
}


@Override
protected Dialog onCreateDialog(int id) {

    switch (id) {
        case DLG_FlAG:
            return createExampleDialog();
        default:
            return null;
    }
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {

    switch (id) {
        case DLG_FlAG:
            // Clear the input box.
            EditText text = (EditText) dialog.findViewById(TEXT_ID_DLG);
            text.setText("");
            break;
    }
}
private Dialog createExampleDialog() {

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("IP address of AP  ");
    final EditText input = new EditText(this);
     input.setId(TEXT_ID_DLG);
     builder.setView(input);

    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int whichButton) {
            String value = input.getText().toString();
            // ips=value ;
            apsInfo.ip4=value ;
           return;
        }
    });

    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
           return;
        }
    });

    return builder.create();
}

}
package allclasses.emuwifimanager;

public class Apsinfo {
		String apstatus = null;
		String apname = null;
		String macaddress = null;
		String level = null;
		public String ip4 = null;
		boolean selected = false;

		public Apsinfo(String apstatus, String apname, String macaddress, String level,
				boolean selected) {
			super();
			this.apstatus = code;
			this.apname = name;
			this.level = level;
			this.macaddress = macaddress;
			this.selected = selected;
		}

		public String getCode() {
			return apstatus;
		}

		public String getName() {
			return apname;
		}
		public String getMacaddress() {
			return macaddress;
		}

		public String getLevel() {
			return level;
		}
		public boolean isSelected() {
			return selected;
		}

		public void setSelected(boolean selected) {
			this.selected = selected;
		}

}
```

Appendix C: EmuMap Class

```
package allclasses.emuwifimanager;
import allclasses.emuwifimanager.R;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.database.Cursor;//SQlite
import android.os.Bundle;

import com.google.android.gms.maps.*;//Maps
import com.google.android.gms.maps.GoogleMap.OnMarkerClickListener;//select marker
import com.google.android.gms.maps.model.BitmapDescriptorFactory;//view image marker
import com.google.android.gms.maps.model.LatLng;//GPS
import com.google.android.gms.maps.model.Marker;//marker object
import com.google.android.gms.maps.model.MarkerOptions;//show information of marker
//import android.content.Context ;
import android.view.Menu;
import android.view.MenuItem;//view normal.satellite,hybrid map
import android.support.v4.app.FragmentActivity;//map frame
public class EmuMap extends FragmentActivity  implements OnMarkerClickListener {
        public GoogleMap emuMap ;
        Marker towerMarker;
        Cursor cursorAPgroupcount ;
        Cursor c1;
        String Dept;
        ApsApplication apsApplication=new ApsApplication();

        @SuppressLint("SdCardPath")
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                 setContentView(R.layout.main);

                apsApplication.apsDataBase=openOrCreateDatabase("myDB1",MODE_PRIVATE
, null);
                 cursorAPgroupcount=               apsApplication.apsDataBase.rawQuery("SELECT
*,count(dept1)  as cnt FROM Accesspointss GROUP BY dept1 ",null);
                 int count= cursorAPgroupcount.getCount();

                        cursorAPgroupcount.moveToFirst();

                        if(cursorAPgroupcount.moveToFirst()){
                                do{


                                        DisplayRecord( );

                                }while (cursorAPgroupcount.moveToNext())            ;

                                }

                        apsApplication.apsDataBase.close();

                        if (ListOfApsInSite.gotoSitemapFlag==1){
                                ListOfApsInSite.gotoSitemapFlag=0;
```

```
            emuMap.moveCamera(CameraUpdateFactory.newLatLng(new
LatLng(ListOfApsInSite.longitudeAp,ListOfApsInSite.longitudeAp)));
                                        emuMap.animateCamera(CameraUpdateFactory.zoomTo(18));
                                    }
                    emuMap.setOnMarkerClickListener(this);
            }

            @Override
            public boolean onCreateOptionsMenu(Menu menu) {
                        getMenuInflater().inflate(R.menu.accesspoint_map, menu);
                        return super.onCreateOptionsMenu(menu);
            }


            private void DisplayRecord( ) {

                        emuMap                          =                          ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map1)).getMap();
               emuMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
               //LatLng  pnt=new LatLng(35.146,33.91);
               LatLng                                                                          pnt=new
LatLng(Double.parseDouble(cursorAPgroupcount.getString(4)),Double.parseDouble(cursorAPgroupc
ount.getString(5)));//35.146,33.91);

               PutMarker(pnt,
cursorAPgroupcount.getString(3),cursorAPgroupcount.getString(6),cursorAPgroupcount.getString(4),
cursorAPgroupcount.getString(5),cursorAPgroupcount.getString(10));

                        emuMap.moveCamera(CameraUpdateFactory.newLatLng(new
LatLng(35.146,33.91)));
               emuMap.animateCamera(CameraUpdateFactory.zoomTo(15));
            }
   public void CopyDB(InputStream inputSream,OutputStream outputStream) throws IOException{

                        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputSream.read(buffer))>0){
            outputStream.write(buffer, 0, length);
        }
        inputSream.close();
        outputStream.close();
  }

            private void PutMarker(LatLng point,String ico,String dept, String lat,String lng,String cnt){
        // Creating an instance of MarkerOptions
        MarkerOptions markerOptions = new MarkerOptions();

        markerOptions.position(point);
       markerOptions.title(dept);//+"(APs="+cnt+")");
       markerOptions.snippet("Lat="+lat+"\n"+"Lng="+lng+"\n");
       markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.opene));

        if                                                                                          (
ico.equalsIgnoreCase("open"))markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawab
le.opene));
        if
(ico.equalsIgnoreCase("close"))markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawa
ble.closee));
```

```
        if
(ico.equalsIgnoreCase("all"))markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable
.alle));
        towerMarker= emuMap.addMarker(markerOptions);

            }

            public void emuMapTypeclk(MenuItem item )
            {
                        CharSequence  it=item.getTitle();

                        if ( it.length()==3) emuMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                        if ( it.length()==9) emuMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
                        if ( it.length()==6) emuMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

            }

            public void menuitemclk(MenuItem item )
            {

                        Intent  intent =new Intent(EmuMap.this,ScanOfAps.class);
                        startActivity(intent);
            }
            @Override
            public boolean onMarkerClick(Marker m4) {
                        Dept=m4.getTitle();
                        ApsApplication.varTempStoreSite=Dept;
                        Intent  intent =new Intent(EmuMap.this,ListOfApsInSite.class);
                        startActivity(intent);
                        return true;
            }

}
package allclasses.emuwifimanager;
import java.util.Calendar;//current date time
import android.app.Application;//to run the first class
import android.database.sqlite.SQLiteDatabase;//database
import android.location.Criteria;//retrieve new GPS
import android.location.Location;//GPS
import android.location.LocationManager;
public class ApsApplication extends Application {
            public SQLiteDatabase apsDataBase;
            public static String varTempStoreSite;
            public static String ipAp;
            public static String [] valuesMibOID=new String [4];
            public static Double currentTime=0.0;
            public static Double ifoutByte=0.0;
            Calendar calm = Calendar.getInstance();
            @Override
            public void onCreate() {

                        super.onCreate();
                        apsDataBase=openOrCreateDatabase("myDB1",MODE_PRIVATE , null);
                        apsDataBase.execSQL("create table if not exists Accesspointss (apid  INTEGER
PRIMARY KEY ,ssid1 VARCHAR,mac1 VARCHAR ,wep1 VARCHAR ,lat1 VARCHAR, lng1
VARCHAR,dept1 VARCHAR,apdate1 VARCHAR,snr1 VARCHAR,ip1 VARCHAR);");

                         LocationManager          locationManager          =          (LocationManager)
getSystemService(LOCATION_SERVICE);
                   Criteria criteria = new Criteria();
```

```
                criteria.setAccuracy(Criteria.NO_REQUIREMENT);
                criteria.setPowerRequirement(Criteria.NO_REQUIREMENT);
                criteria.setCostAllowed(false);

                String bestProvider = locationManager.getBestProvider(criteria, false);
                Location location = locationManager.getLastKnownLocation(bestProvider);

        }
}
```

Appendix D: SiteSearching Class

```
package allclasses.emuwifimanager;
import allclasses.emuwifimanager.R;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;//SQLite
import android.os.Bundle;
import android.text.Editable;//enter site
import android.text.TextWatcher;//combo box
import android.view.View;// view
import android.widget.AdapterView;
import android.widget.ArrayAdapter;// array adapter
import android.widget.EditText;// Edit text
import android.widget.ListView;

import android.widget.AdapterView.OnItemClickListener;

public class SiteSearching extends Activity
{

        // List view
        private ListView listviewSites;
         ApsApplication apsapplication;

        // Listview Adapter
        ArrayAdapter<String> adapter;

        // Search EditText
        EditText inputSearch;


         String siteName;
   @Override
   public void onCreate(Bundle savedInstanceState)
   {

       super.onCreate(savedInstanceState);
       setContentView(R.layout.viewsite);
       apsapplication=new ApsApplication();
                apsapplication.apsDataBase=openOrCreateDatabase("myDB1",MODE_PRIVATE ,
null);

                // Listview Data
```

```
             Cursor  c=  apsapplication.apsDataBase.rawQuery("SELECT  DISTINCT  dept1  FROM
Accesspointss ",null);
            String[] site=new String[c.getCount()];
                              c.moveToFirst();
                              if(c.moveToFirst()){
                                        int i=0;
                                  do{
                                        site[i]=c.getString(c.getColumnIndex("dept1"));
                                        i++;
                                        }while (c.moveToNext())  ;

                                        }

                              apsapplication.apsDataBase.close();

       listviewSites = (ListView) findViewById(R.id.list_view);
       inputSearch = (EditText) findViewById(R.id.inputSearch);

       // Adding items to listview
       adapter = new ArrayAdapter<String>(this, R.layout.viewsiteinfo, R.id.city_name, site);
       listviewSites.setAdapter(adapter);


       listviewSites.setOnItemClickListener(new OnItemClickListener()
                   {

                        public  void  onItemClick(AdapterView<?>  parent,  View  view,  int  position,
long id)
                                    {
                            siteName =(String) parent.getItemAtPosition(position);
                                              ApsApplication.varTempStoreSite=siteName;
                            Intent  intent =new Intent(SiteSearching.this,ListOfApsInSite.class);
                                              startActivity(intent);
                        }
                   });


       /**
        * Search Filter Enable
        * */

       inputSearch.addTextChangedListener(new TextWatcher()
       {

                            @Override
                            public void onTextChanged(CharSequence cs, int arg1, int arg2, int arg3)
                            {
                                  // When user changed the Text
                                  SiteSearching.this.adapter.getFilter().filter(cs);
                            }

                            @Override  //because TextWatcher
                   public void beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3)
                            {


                            }

                            @Override//because TextWatcher
                            public void afterTextChanged(Editable arg0)
                        {

                        }
                });
    }}
```

Appendix E: ListOfApsInSite Class

```
package allclasses.emuwifimanager;

import java.util.ArrayList;

import allclasses.emuwifimanager.R;
import android.app.Activity;//activity
import android.app.AlertDialog;//delete

import android.content.Context;
import android.content.DialogInterface;//delete
import android.content.Intent;//call activity
import android.database.Cursor;//data base
import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;
//import android.util.Log;
import android.view.LayoutInflater;
import android.view.MotionEvent;//touch IP to go to the snmp
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;//event

import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;//delete and view on map
import android.widget.CheckBox;//select to delete
import android.widget.ListView;
import android.widget.TextView;//number of APs
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class ListOfApsInSite extends Activity {

        CustomAdapter dataAdapterOfAPs = null;
        ApsApplication apsApplication;
        TextView txtSiteName_APsCount;
        String countAPsOfSite;
        public static int gotoSitemapFlag;
        public static double latitudeAP;
        public static double longitudeAp;
        ListView lstViewOfAPsStore;
        AlertDialog.Builder alertBuilderDelete;
        Button deleteButton;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.viewaps);
                alertBuilderDelete = new AlertDialog.Builder(this);
                apsApplication = new ApsApplication();
                apsApplication.apsDataBase = openOrCreateDatabase("myDB1",
                            MODE_PRIVATE, null);
                int oo = SQLiteDatabase.OPEN_READWRITE;
                Cursor c = apsApplication.apsDataBase.rawQuery(
        "SELECT apid,ssid1,lat1,lng1,snr1 FROM Accesspointss where dept1='"
                                    + ApsApplication.varTempStoreSite + "' ", null);
                if (c.getCount() != 0) {
                        countAPsOfSite = "" + c.getCount() + " " + "Touch ip for SNMP";
                        displayListView();
                }
                checkButtonClick();

        }

        private void displayListView() {

                txtSiteName_APsCount = (TextView) findViewById(R.id.siteSelected);

                txtSiteName_APsCount.setText(ApsApplication.varTempStoreSite + " "
                            + "APs=" + countAPsOfSite);
                Cursor c = apsApplication.apsDataBase.rawQuery(
        "SELECT apid,ssid1,lat1,lng1,snr1,ip1 FROM Accesspointss where dept1='"
                            + ApsApplication.varTempStoreSite + "' ", null);
                String[] ssid = new String[c.getCount()];
                String[] snr = new String[c.getCount()];
                c.moveToFirst();

                latitudeAP = Double.parseDouble(c.getString(2));
                longitudeAp = Double.parseDouble(c.getString(3));
                c.moveToFirst();
                if (c.moveToFirst()) {
                        int i = 0;
                        do {
                                ssid[i] = c.getString(c.getColumnIndex("ssid1"));
                                snr[i] = c.getString(c.getColumnIndex("snr1")) + " "
                                                + c.getString(c.getColumnIndex("ip1"));
                                i++;
                        } while (c.moveToNext());

                }

                apsApplication.apsDataBase.close();

                // Array list of Aps
                ArrayList<ListOfApsinfo> listOfApsinfoArray = new ArrayList<ListOfApsinfo>();
                for (int i = 0; i < snr.length; i++) {
                        ListOfApsinfo listApsinfo = new ListOfApsinfo(snr[i], ssid[i],
                                    false);
                        listOfApsinfoArray.add(listApsinfo);
                }

                // create an ArrayAdaptar from the String Array
                dataAdapterOfAPs = new CustomAdapter(this, R.layout.viewapsinfo,
                            listOfApsinfoArray);
                lstViewOfAPsStore = (ListView) findViewById(R.id.listView1);
                // Assign adapter to ListView
                lstViewOfAPsStore.setAdapter(dataAdapterOfAPs);

                lstViewOfAPsStore.setOnItemClickListener(new OnItemClickListener() {

                        public void onItemClick(AdapterView<?> parent, View view,
                                        int position, long id) {

                              }
                    });
          }

          private class CustomAdapter extends ArrayAdapter<ListOfApsinfo> {

                    private ArrayList<ListOfApsinfo> listApsArray;

                    public CustomAdapter(Context context, int textViewResourceId,

                    ArrayList<ListOfApsinfo> stateList) {
                              super(context, textViewResourceId, stateList);
                              this.listApsArray = new ArrayList<ListOfApsinfo>();
                              this.listApsArray.addAll(stateList);
                    }

                    private class ViewHolderChkBoxSsidSNRIP {
                              TextView ssidSNRIP;
                              CheckBox nameChkBox;
                    }

                    @Override
                    public View getView(int position, View convertView, ViewGroup parent) {

                              ViewHolderChkBoxSsidSNRIP apholder = null;
                              if (convertView == null) {

                                        LayoutInflater            vi            =            (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                                        convertView = vi.inflate(R.layout.viewapsinfo, null);
                                        apholder = new ViewHolderChkBoxSsidSNRIP();
                                        apholder.ssidSNRIP = (TextView) convertView
                                                  .findViewById(R.id.code);
                                        apholder.nameChkBox = (CheckBox) convertView
                                                  .findViewById(R.id.checkBox1);
                                        convertView.setTag(apholder);

                                        apholder.nameChkBox
          .setOnClickListener(new View.OnClickListener() {
                                                            public void onClick(View v) {
                                                                      deleteButton.setEnabled(true);
                                                                      CheckBox  cb  =  (CheckBox)
v;
                                                                      ListOfApsinfo      _state      =
(ListOfApsinfo) cb
                                                                                          .getTag();

          Toast.makeText(getApplicationContext(), cb.getText()

          + " is " + cb.isChecked(),

          Toast.LENGTH_LONG).show();


          _state.setSelected(cb.isChecked());
                                                            }
                                                  });
```

```
                                apholder.ssidSNRIP.setOnTouchListener(new
View.OnTouchListener() {

                                        @Override
                                        public boolean onTouch(View v, MotionEvent event) {
                                                int eventaction = event.getAction();

switch (eventaction) {
case MotionEvent.ACTION_UP:
// finger leaves the screen
TextView ipc = (TextView) v;
ApsApplication.ipAp = ipc.getText().toString()
                .substring(4);
                Intent intent = new Intent(ListOfApsInSite.this,SnmpSetting.class);
                                                        startActivity(intent);
                                                        break;
                                                }
                                                return true;
                                        }

                                });

                        } else {
                                apholder        =       (ViewHolderChkBoxSsidSNRIP)
convertView.getTag();
                        }

                        ListOfApsinfo aplist = listApsArray.get(position);

                        apholder .ssidSNRIP.setText(aplist.getCode());
                        apholder .nameChkBox.setText(aplist.getName());
                        apholder .nameChkBox.setChecked(aplist.isSelected());

                        apholder.nameChkBox.setTag(aplist);

                        return convertView;
                }

        }

        private void checkButtonClick() {

                deleteButton = (Button) findViewById(R.id.findSelected);
                deleteButton.setEnabled(false);

                deleteButton.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {

                                StringBuffer textResponse = new StringBuffer();
                                textResponse.append("The following were selected...\n");
                                apsApplication = new ApsApplication();
                                apsApplication.apsDataBase = openOrCreateDatabase("myDB1",
                                                MODE_PRIVATE, null);
                                int rw = SQLiteDatabase.OPEN_READWRITE;
                                final       ArrayList<ListOfApsinfo>       stateList       =
dataAdapterOfAPs.listApsArray;
                                alertBuilderDelete.setTitle("Confirm");
                                alertBuilderDelete.setMessage("Are you sure for deleting?");
```

```
                                        alertBuilderDelete.setPositiveButton("YES",
                                                    new DialogInterface.OnClickListener() {

                                                         public void onClick(DialogInterface
dialog,int which) {
                                                              // Do nothing but close the
dialog
for (int i = 0; i < stateList.size(); i++) {

ListOfApsinfo aplist = stateList.get(i);

if ((aplist.isSelected())) {

     Toast.makeText(getApplicationContext(),

     "deleted: " + aplist.getName(),

     Toast.LENGTH_LONG).show();

     apsApplication.apsDataBase

     .execSQL("DELETE FROM Accesspointss WHERE ssid1='"

               + aplist.getName() + "'");
                                                                   }
                                                              }

     apsApplication.apsDataBase.close();

                                                              dialog.dismiss();

                                                              finish();
                                                              Intent intent = new Intent(

     ListOfApsInSite.this,

     ListOfApsInSite.class);
                                                              startActivity(intent);

                                                         }

                                                    });

                                        alertBuilderDelete.setNegativeButton("NO",
                                                    new DialogInterface.OnClickListener() {

                                                         @Override
                                                         public   void   onClick(DialogInterface
dialog,
                                                                   int which) {
                                                              // Do nothing
                                                              dialog.dismiss();
                                                              finish();
                                                              Intent intent = new Intent(

     ListOfApsInSite.this,

     ListOfApsInSite.class);
                                                              startActivity(intent);
```

```
							}
						});

					AlertDialog alert = alertBuilderDelete.create();
					alert.show();
					dataAdapterOfAPs.notifyDataSetChanged();
					lstViewOfAPsStore.invalidateViews();

				}
			});
		}

		public void viewsiteonmap(View v) {
			gotoSitemapFlag = 1;
			Intent intent = new Intent(this, allclasses.emuwifimanager.EmuMap.class);
			startActivity(intent);

		}

}

package allclasses.emuwifimanager;

public class ListOfApsinfo {

		String ssidSNRIP = null;
		String nameChkBox = null;
		boolean selected = false;

		public ListOfApsinfo(String ssidSNRIP, String nameChkBox, boolean selectedChkBox) {
			super();
			this. ssidSNRIP = ssidSNRIP;
			this. nameChkBox = nameChkBox;
			this.selected = selectedChkBox;
		}

		public String getCode() {
			return ssidSNRIP;
		}

		public void setCode(String code) {
			this. ssidSNRIP = code;
		}

		public String getName() {
			return nameChkBox;
		}

		public void setName(String name) {
			this. nameChkBox = name;
		}

		public boolean isSelected() {
			return selected;
		}

		public void setSelected(boolean select) {
			this.selected = select;
		}
```

```
}
```

Appendix F: SnmpSetting Class

```
package allclasses.emuwifimanager;
import java.io.File;//get file
import java.io.FilenameFilter;
import java.util.ArrayList;
import allclasses.emuwifimanager.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.EditText;
public class SnmpSetting extends Activity {
	EditText edittxtMibFile;
	EditText edittxtIP;
	EditText edittxtPort;
	EditText edittxtSnmpVersion;
	EditText edittxtCommunity;
	EditText edittxtTimeOut;
	AlertDialog levelDialog;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.settingsnmp);
		    edittxtMibFile=(EditText)findViewById(R.id.mibfile);
		    edittxtIP=(EditText)findViewById(R.id.ipee);
		   //ip1.setText(ApsApplication.ipap);
		    edittxtPort=(EditText)findViewById(R.id.portee);
		    edittxtSnmpVersion=(EditText)findViewById(R.id.versionee);
		    edittxtCommunity=(EditText)findViewById(R.id.commnee);
		    edittxtTimeOut=(EditText)findViewById(R.id.timeee);

		    edittxtMibFile.setOnTouchListener ( new View.OnTouchListener() {

				@Override
				public boolean onTouch(View v, MotionEvent event) {
					int eventaction = event.getAction();

				    switch (eventaction) {


				        case MotionEvent.ACTION_UP:
				            // finger leaves the screen
				             viewMibFiles();
				             break;
				    }



					return true;
				}
```

```
                        });

            }

            public void viewMibFiles (){

                        ArrayList<String> files=new ArrayList<String>();

                        File home = new File("/storage/sdcard0/mibs/");

                         if (home.listFiles(new FileExtensionFilter()).length > 0) {
                              for (File file : home.listFiles(new FileExtensionFilter())) {

                                 files.add(file.getName().substring(0,   (file.getName().length()  )));


                              }
                           }
                         final String [] items = files.toArray(new String[files.size()]);

                                    // Strings to Show In Dialog with Radio Buttons


                                    // Creating and Building the Dialog
                                    AlertDialog.Builder builder = new AlertDialog.Builder(this);
                                    builder.setTitle("Mibs");

                                    builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {

                                       @Override
                                       public void onClick(DialogInterface dialog, int whichButton) {

                                          // ips=value ;

                                          return;
                                       }
                                    });


                                    builder.setSingleChoiceItems(items,                    -1,                    new
DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int item) {

                                              edittxtMibFile.setText(items[item]);

                                       }
                                    });
                                    levelDialog = builder.create();
                                    levelDialog.show();

            }


             class FileExtensionFilter implements FilenameFilter {
                         public boolean accept(File dir, String name) {
                            return           (name.endsWith(".mib")          ||          name.endsWith(".txt")||
name.endsWith(".my"));
                         }
```

```
                    }

public void veiwmibtreeclk(View view ) {
            String mfs=edittxtMibFile.getText().toString();
            String ip11=edittxtIP.getText().toString();
            String port11=edittxtPort.getText().toString();
            String ver11=edittxtSnmpVersion.getText().toString();
            String comn11=edittxtCommunity.getText().toString();
            String time11=edittxtTimeOut.getText().toString();
               Intent  intent =new Intent( this,SnmpTreeMake.class);
                        intent.putExtra("MIB_FILE", mfs);
                        intent.putExtra("IP", ip11);
                        intent.putExtra("PORT", port11);
                        intent.putExtra("VER", ver11);
                        intent.putExtra("COMMN", comn11);
                        intent.putExtra("TIME", time11);
                        startActivity(intent);

            }

public void networkutilizeclk(View view ) {

            allclasses.emuwifimanager.NetworkUtilization.mip=edittxtIP.getText().toString();
            allclasses.emuwifimanager.NetworkUtilization.mport=edittxtPort.getText().toString();
            allclasses.emuwifimanager.NetworkUtilization.mver=edittxtSnmpVersion.getText().toString
();
            allclasses.emuwifimanager.NetworkUtilization.mcommn=edittxtCommunity.getText().toStri
ng();
            allclasses.emuwifimanager.NetworkUtilization.mtime=edittxtTimeOut.getText().toString();

            Intent  intent =new Intent( this,NetworkUtilization.class);

                        startActivity(intent);

            }

}




package  allclasses.emuwifimanager;
import  java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.List;
import java.util.Vector;

import allclasses.emuwifimanager.R;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.SparseArray;

import android.widget.ExpandableListView;
import com.adventnet.snmp.mibs.*;
import com.adventnet.snmp.snmp2.SnmpOID;
public class SnmpTreeMake extends Activity {
            MibOperations mibOps   = new MibOperations(); ;
            Context context;
             Group Grpnew ;
             MibTreeListAdapter newadapter; //= new MibTreeListAdapter(this,groups);
             String  mfile;
             String  mip;
             String  mport;
             String  mver;
             String  mcommn;
             String  mtime;
            ExpandableListView listView;
            SnmpGet snmpget;
            String oidnum;
            //create tree
             SparseArray<Group> groups = new SparseArray<Group>();//tree
                        @Override
             protected void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.activity_main);
               context = this;
               listView = (ExpandableListView) findViewById(R.id.listexpand);

               System.setProperty("java.net.preferIPv4Stack", "true");
                        System.setProperty("java.net.preferIPv6Addresses", "false");

                         mfile =getIntent().getStringExtra("MIB_FILE")  ;
                         mip =getIntent().getStringExtra("IP")  ;
                         mip=mip.trim( );
                         mport =getIntent().getStringExtra("PORT")  ;
                         mver =getIntent().getStringExtra("VER")  ;
                         mcommn =getIntent().getStringExtra("COMMN")  ;
                         mtime =getIntent().getStringExtra("TIME")  ;

               try {

                        mibOps.loadMibModule("/storage/sdcard0/mibs/"+mfile);

                                    mibOps.setLoadFromCompiledMibs(true);
                        } catch (FileNotFoundException e) {

                                    e.printStackTrace();
                        } catch (MibException e) {

                                    e.printStackTrace();
                        } catch (IOException e) {
                                    e.printStackTrace();
                        }
              // createData();
               try {
                                    TreeView();
                        } catch (Exception e) {
                                    e.printStackTrace();
                        }

              mAsyncTask.execute();
```

```
                try {
                                        Thread.sleep(180);
                            } catch (InterruptedException e) {
                                        e.printStackTrace();
                            }


    } //end of  onCreate


            @SuppressWarnings("rawtypes")//this for Enumeration en  and this for Vector v
            @SuppressLint("SdCardPath")
            public void TreeView( ) throws Exception  {


                            // MibModule module = mibOps.getMibModule("RFC1213-MIB");
                      boolean cisco1= mibOps.getModuleNameDefinition(mfile).contains("CISCO");
                            MibModule              module              =              mibOps.getMibModule(
mibOps.getModuleNameDefinition(mfile));


                            List<String> oids = new ArrayList<String>();

                   Enumeration en;
                    MibNode node;
                     SnmpOID snmpoid;
                      en= module.getDefinedNodes();
                    List<String> Treeoid = new ArrayList<String>();

                    while ( en.hasMoreElements()){
                              String NodeN=en.nextElement().toString();
                             // Toast.makeText(this,""+NodeN,Toast.LENGTH_LONG).show();
                              snmpoid = mibOps.getSnmpOID(NodeN);
                              node = mibOps.getMibNode(snmpoid);
                             if (node!=null){
                                       String s=node.getOIDString();

                                       s=s.substring(1);//to begin from second char becuse first is . dot
                   s=s.replace('.', '>');
                                      Treeoid.add(s);
                                      }
                            }
                    Collections.sort(Treeoid);

                     List<String> list1=new ArrayList<String>();
                    for (String path : Treeoid) {

                             path=path.replace('>', '.') ;

                             snmpoid = mibOps.getSnmpOID(path);
                             node = mibOps.getMibNode(snmpoid);
                              if (cisco1){

                                        Vector vv=  node.getChildList();
                                        if(vv.size()==0){

                     if (node.getParent()!=null){
                            list1.add(node.getParent().toString());
                        }
                     }
```

```
                        }
                        else  {
                          if(node.isLeaf()){
                                list1.add(node.getParent().toString());

        }
                        }
                }
                Collections.sort(list1);
                String li=list1.get(0);
                int key=0;
                Group group;
                int cnt;
                for(  cnt=1;cnt<list1.size();cnt++){

                        if(list1.get(cnt).equals(li))
                        {
                                continue;
                         }
                       else {

                                group = new Group(li);
                                node = mibOps.getMibNode(li);

                               Vector v=  node.getChildList();
                                for (int v1=0;v1<v.size();v1++)
                                    {
                                          if                                        (cisco1){
oidnum=mibOps.getSnmpOID(v.get(v1).toString()).toString();}
                                         else{

oidnum=mibOps.getSnmpOID(v.get(v1).toString()).toString()+".1";
                                           }
                                  group.children.add(v.get(v1).toString() +"\n"+oidnum );


                                    }

                                oids.add(li);
                                  li=list1.get(cnt);
                          }
                        groups.append(key, group);
                        key++;

                        }


                ///// get last value of leave of snmp
                li=list1.get(cnt-1);
                group = new Group(li);
                node = mibOps.getMibNode(li);

              Vector v=  node.getChildList();
               for (int v1=0;v1<v.size();v1++)
                    {

                       if                                                   (cisco1){
oidnum=mibOps.getSnmpOID(v.get(v1).toString()).toString();}
                      else{
                              oidnum=mibOps.getSnmpOID(v.get(v1).toString()).toString()+".0";
                               }



                         group.children.add(v.get(v1).toString() +"\n"+oidnum );


                          }


                       oids.add(li);

                       groups.append(key, group);

                       MibTreeListAdapter adapter = new MibTreeListAdapter(this,groups);
                              listView.setAdapter(adapter);

   }

         AsyncTask<Void, Void, Void> mAsyncTask = new AsyncTask<Void, Void, Void>() {
                    @Override
                    protected Void doInBackground(Void... params) {
                            SnmpGet          snmpget          =new          SnmpGet(
mip,mport,mver,mcommn,mtime);
                            String oidnum2;
                            String[] oidnum;
                            //String s1="1.3.6.1.2.1.2.2.1.1.33";
                           // String s1=".1.3.6.1.2.1.2.2.1.2";
                          for (int m=0;m<groups.size()  ;m++){

                           for(int o=0;o<groups.get(m).children.size();o++){
                                oidnum=    groups.get(m).children.get(o).toString().split(
System.getProperty("line.separator"));
                                 oidnum2=oidnum[1];

                                      try {
                                           //   snmpget.sendSnmpRequest(s1);
                                            //
snmpget.sendSnmpRequest(".1.3.6.1.2.1.1.1.0");

                                            snmpget.sendSnmpRequest(oidnum2);

                                      } catch (Exception e) {

                                           e.printStackTrace();
                                       }

groups.get(m).children.set(o,groups.get(m).children.get(o)+"\n"+snmpget.res );
                          }
                          }
                                                            return null;
                    }


          };

         }
```

```
package allclasses.emuwifimanager;
import org.snmp4j.CommunityTarget;
import org.snmp4j.PDU;
import org.snmp4j.Snmp;
import org.snmp4j.TransportMapping;
import org.snmp4j.event.ResponseEvent;
import org.snmp4j.mp.SnmpConstants;
import org.snmp4j.smi.OID;
import org.snmp4j.smi.OctetString;
import org.snmp4j.smi.UdpAddress;
import org.snmp4j.smi.VariableBinding;
import org.snmp4j.transport.DefaultUdpTransportMapping;

import android.content.Context;

public class SnmpGet   {

                        private static String ipAddress  ;
                        private static   String port ;
      // command to request from Server

                         private static   int SNMP_VERSION ;
                        private static   String community ;
                        private static   String time1 ;
                        public static Snmp snmp;
                        public static CommunityTarget comtarget;
                        static PDU pdu;
                        static OID oid;
                        PDU responsePDU;
                        public  String res;
                        Context context;
                        public SnmpGet (String ipp,String portt,String vert,String commnt,String timet){
                                    ipAddress=ipp;
                                    port=portt;
                                    if (vert.equals("v1")) SNMP_VERSION= SnmpConstants.version1;
                                    if (vert.equals("v2")) SNMP_VERSION= SnmpConstants.version2c;
                                    if (vert.equals("v3")) SNMP_VERSION= SnmpConstants.version3;

                                    community= commnt;
                                    time1= timet;
                        }

                        public   void  sendSnmpRequest(String cmd) throws Exception {

                                    // Create TransportMapping// and Listen//
                                    //Toast.makeText(context,"location                                    change",
Toast.LENGTH_LONG).show();
                                    TransportMapping<UdpAddress>            transport            =            new
DefaultUdpTransportMapping();
                                    transport.listen();
                        PDU pdu = new PDU();
                                    pdu.add(new VariableBinding(new OID(cmd)));
                           pdu.setType(PDU.GETNEXT);
                                      snmp = new Snmp(transport);
                                    // send the PDU
                                    ResponseEvent response = snmp.get(pdu, getTarget());
                                    // Process //Agent Response
                                    if (response != null) {
                                                // extract the response PDU (null if timed out)
                                                  responsePDU = response.getResponse();
```

```
                                        // extract the address //used by //the agent to send the response:

                                         // get pdu
                                        if (responsePDU != null) {
                                                int errorStatus = responsePDU.getErrorStatus();
                                                if (errorStatus == PDU.noError) {
                                                        int
resat=responsePDU.getVariableBindings().toString().indexOf("=");
                        res=responsePDU.getVariableBindings().toString().substring(resat+1);
                        res=res.substring(0,  res.length()-1);// to delete ] from end

                                                }
                                        }
                                }
                                snmp.close();

                        }


                        private CommunityTarget getTarget() {
                                // Create Target Address object
                                                CommunityTarget comtarget = new CommunityTarget();
                                                comtarget.setCommunity(new OctetString(community));
                                                comtarget.setVersion(SNMP_VERSION);
                                                comtarget.setAddress(new  UdpAddress(ipAddress  +  "/"
+ port));
                                                comtarget.setRetries(2);
                                                comtarget.setTimeout(1000*Integer.parseInt(time1));
                          return comtarget;
                        }

                }

package allclasses.emuwifimanager;
import java.util.ArrayList;// array list  for children
import java.util.List;//list  for children
public class Group {

                public String childstring;
                  public final List<String> children = new ArrayList<String>();

                  public Group(String stringg) {
                    this.childstring = stringg;
                  }

}

package allclasses.emuwifimanager;
import allclasses.emuwifimanager.R;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import android.os.AsyncTask;
import java.util.Calendar;
public class NetworkUtilization extends Activity
{
```

```
String oidvalue;
public int k=0;
public int value=1;
int cntsnmp=0;
String ifSpeed,ifoutOctets;
Calendar cal = Calendar.getInstance();
  public static String  mip;
  public static String  mport;
  public static String  mver;
  public static String  mcommn;
  public static String  mtime;
  SnmpGet snmpget ;
  Double interval=0.0;
@Override
public void onCreate(Bundle savedInstanceState)
{

    super.onCreate(savedInstanceState);
    setContentView(R.layout.chart);
    TextView titel= (TextView)findViewById(R.id.textView1);
    titel.setTextColor(Color.MAGENTA);
    titel.setTextSize(18.0f);
    TextView ifs1= (TextView)findViewById(R.id.textView5);
    ifs1.setTextColor(Color.RED);
    ifs1.setTextSize(16.0f);
    TextView ifout1= (TextView)findViewById(R.id.textView8);
    ifout1.setTextColor(Color.RED);
    ifout1.setTextSize(16.0f);
    TextView OutputUtilization2= (TextView)findViewById(R.id.textView10);
    OutputUtilization2.setTextColor(Color.RED);
    OutputUtilization2.setTextSize(16.0f);
    TextView ifs= (TextView)findViewById(R.id.ifSpeed);
    ifs.setTextColor(Color.BLUE);
    ifs.setTextSize(14.0f);
    TextView ifout= (TextView)findViewById(R.id.ifoutOctets);
    ifout.setTextColor(Color.BLUE);
    ifout.setTextSize(14.0f);
    TextView OutputUtilization1= (TextView)findViewById(R.id.OutputUtilization);
    OutputUtilization1.setTextColor(Color.BLUE);
    OutputUtilization1.setTextSize(14.0f);
    System.setProperty("java.net.preferIPv4Stack", "true");
                    System.setProperty("java.net.preferIPv6Addresses", "false");
                    snmpget =new SnmpGet( mip,mport,mver,mcommn,mtime);
                    mAsyncTask.execute();
            try {
                                Thread.sleep(200);
                        } catch (InterruptedException e) {

                                e.printStackTrace();
                        }

            ifs.setText(ifSpeed);
            ifout.setText(ifoutOctets);
            Double s=0.0;
            Double o=0.0;
            int sec=0;
            interval=Double.parseDouble(""+cal.getTime().getSeconds()) ;
            if (ifSpeed !=null){
             s=Double.parseDouble(ifSpeed );    //parseInt(ifs.getText().toString());
```

```
                o=Double.parseDouble(ifoutOctets);

               }

               if (ApsApplication.ifoutByte==0.0){
                         ApsApplication.ifoutByte=o;
                         ApsApplication.currentTime=interval;

                }

                else{
                         Double up1=(o-ApsApplication.ifoutByte)*8*100;
                         Double do1=(s*(interval-ApsApplication.currentTime));
                         Double diff=interval-ApsApplication.currentTime;
                         OutputUtilization1.setText( up1/do1 +"\n% for time interval="+diff+" sec"  );
                         ApsApplication.ifoutByte=o;
                         ApsApplication.currentTime=interval;
                }

               Toast.makeText(this,mip+mport+mver+mcommn+mtime+"            "+            ifoutOctets+"
"+cal.getTime().getSeconds() , Toast.LENGTH_LONG).show();

   }
   AsyncTask<Void, Void, Void> mAsyncTask = new AsyncTask<Void, Void, Void>() {
                   @Override
                   protected Void doInBackground(Void... params) {


                            try {
                                          snmpget.sendSnmpRequest(".1.3.6.1.2.1.2.2.1.5.13");
                                          ifSpeed=snmpget.res ;
                                          snmpget.sendSnmpRequest(".1.3.6.1.2.1.2.2.1.16.13");
                                          ifoutOctets=snmpget.res ;

                            } catch (Exception e) {

                                                         e.printStackTrace();
                                               }


                                                               return null;
                   }


  };

  public void clkrefresh(View view ) {

              finish();
              Intent       mainIntent       =       new       Intent().setClass(       NetworkUtilization.this,
NetworkUtilization.class);
     startActivity(mainIntent);
             }
}

package allclasses.emuwifimanager;
import allclasses.emuwifimanager.R;
import android.app.Activity;//activity
import android.util.SparseArray;//array
import android.view.LayoutInflater;//inflater
import android.view.View;//view
import android.view.ViewGroup;//group
import android.widget.BaseExpandableListAdapter;//adapter
import android.widget.CheckedTextView;//chk box
import android.widget.TextView;//text view

public class MibTreeListAdapter     extends BaseExpandableListAdapter {

        private final SparseArray<Group> groups;
        public LayoutInflater inflater;
        public Activity activity;
        public MibTreeListAdapter(Activity acti, SparseArray<Group> groups) {
          activity = acti;
          this.groups = groups;
          inflater = acti.getLayoutInflater();

        }

        @Override
        public Object getChild(int groupPos, int childPos) {//get child

                return groups.get(groupPos).children.get(childPos);

        }

        @Override
        public long getChildId(int groupPos, int childPos) {//get child pos
          return 0;
        }

        @Override
        public View getChildView(int groupPos, final int childPos,
            boolean isLastChild, View convertView, ViewGroup parent) {
          final String children = (String) getChild(groupPos, childPos);
          TextView text = null;
          if (convertView == null) {
            convertView = inflater.inflate(R.layout.listrow_details, null);
          }
          text = (TextView) convertView.findViewById(R.id.textLeaf);

          text.setText(children);

          return convertView;
        }

        @Override
        public int getChildrenCount(int groupPos) {
          return groups.get(groupPos).children.size();
        }

        @Override
        public Object getGroup(int groupPos) {
          return groups.get(groupPos);
        }

        @Override
        public int getGroupCount() {
          return groups.size();
```

```
            }

            @Override
            public void onGroupCollapsed(int groupPos) {
              super.onGroupCollapsed(groupPos);

            }

            @Override
            public void onGroupExpanded(int groupPos) {
              super.onGroupExpanded(groupPos);


            }

            @Override
            public long getGroupId(int groupPos) {
              return 0;
            }

            @Override
            public View getGroupView(int groupPos, boolean isExpanded,
                View convertView, ViewGroup parent) {
              if (convertView == null) {
                convertView = inflater.inflate(R.layout.listrow_group, null);
              }
              Group group = (Group) getGroup(groupPos);
              ((CheckedTextView) convertView).setText(group.string);
              ((CheckedTextView) convertView).setChecked(isExpanded);
              return convertView;
            }

            @Override
            public boolean hasStableIds() {
              return false;
            }

            @Override
            public boolean isChildSelectable(int groupPos, int childPos) {
              return false;
            }

}
```

Appendix G: Monkey Testing Report

```
:Monkey: seed=0 count=500

:AllowPackage: allclasses.emuwifimanager

:IncludeCategory: android.intent.category.LAUNCHER

:IncludeCategory: android.intent.category.MONKEY

// Event percentages:

//   0: 15.0%
//   1: 10.0%

//   2: 2.0%

//   3: 15.0%

//   4: -0.0%

//   5: 25.0%

//   6: 15.0%

//   7: 2.0%

//   8: 2.0%

//   9: 1.0%

//   10: 13.0%

:Switch:
#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFla
gs=0x10200000;component=allclasses.emuwifimanager/.MainMenu;end

   // Allowing start of Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] cmp=allclasses.emuwifimanager/.MainMenu } in package
allclasses.emuwifimanager

:Sending Flip keyboardOpen=false

:Sending Touch (ACTION_DOWN): 0:(313.0,367.0)

:Sending Touch (ACTION_UP): 0:(313.19876,367.49512)

   // Allowing start of Intent { cmp=allclasses.emuwifimanager/.ScanOfAps } in package
allclasses.emuwifimanager

:Sending Touch (ACTION_DOWN): 0:(427.0,724.0)

:Sending Touch (ACTION_UP): 0:(432.7109,722.9021)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,-1.0)

   // Rejecting start of Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] cmp=com.android.browser/.BrowserActivity } in package
com.android.browser

:Sending Trackball (ACTION_MOVE): 0:(1.0,-3.0)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,-2.0)

:Sending Trackball (ACTION_UP): 0:(0.0,0.0)

:Sending Touch (ACTION_DOWN): 0:(257.0,143.0)

:Sending Touch (ACTION_UP): 0:(248.47726,134.0601)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,-2.0)
```

```
:Sending Touch (ACTION_DOWN): 0:(353.0,293.0)

:Sending Touch (ACTION_UP): 0:(367.02255,292.70322)

:Sending Touch (ACTION_DOWN): 0:(460.0,368.0)

:Sending Touch (ACTION_UP): 0:(478.2656,423.67996)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,-4.0)

    //[calendar_time:2014-05-18 08:35:36.165  system_uptime:51974053]

    // Sending event #100

    // Rejecting start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.HOME] cmp=com.sec.android.app.launcher/com.android.launcher2.Launcher } in package com.sec.android.app.launcher

:Sending Touch (ACTION_DOWN): 0:(57.0,301.0)

:Sending Touch (ACTION_UP): 0:(37.1931,292.35184)

:Sending Touch (ACTION_DOWN): 0:(314.0,689.0)

:Sending Touch (ACTION_UP): 0:(311.58575,682.51685)

:Sending Touch (ACTION_DOWN): 0:(270.0,402.0)

:Sending Touch (ACTION_UP): 0:(292.0574,426.04135)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,-1.0)

:Sending Touch (ACTION_DOWN): 0:(87.0,357.0)

:Sending Touch (ACTION_UP): 0:(86.42902,362.98502)

:Sending Touch (ACTION_DOWN): 0:(180.0,667.0)

:Sending Touch (ACTION_UP): 0:(174.44862,618.22144)

:Sending Touch (ACTION_DOWN): 0:(321.0,271.0)

:Sending Touch (ACTION_UP): 0:(323.4723,269.23087)

:Sending Touch (ACTION_DOWN): 0:(263.0,629.0)

:Sending Touch (ACTION_UP): 0:(260.67447,628.6886)

:Sending Trackball (ACTION_MOVE): 0:(-4.0,3.0)

:Sending Touch (ACTION_DOWN): 0:(455.0,427.0)

:Sending Touch (ACTION_UP): 0:(473.70074,509.83344)

:Sending Touch (ACTION_DOWN): 0:(220.0,734.0)

:Sending Touch (ACTION_UP): 0:(227.5121,719.42615)

:Sending Touch (ACTION_DOWN): 0:(60.0,488.0)
```

```
:Sending Touch (ACTION_UP): 0:(59.627926,492.37195)

:Sending Touch (ACTION_DOWN): 0:(17.0,377.0)

    //[calendar_time:2014-05-18 08:35:37.726  system_uptime:51975589]

    // Sending event #200

:Sending Touch (ACTION_UP): 0:(0.0,365.55975)

:Sending Trackball (ACTION_MOVE): 0:(-3.0,2.0)

:Sending Trackball (ACTION_MOVE): 0:(4.0,4.0)

:Sending Touch (ACTION_DOWN): 0:(54.0,538.0)

:Sending Touch (ACTION_UP): 0:(70.23189,528.46063)

:Sending Touch (ACTION_DOWN): 0:(412.0,777.0)

:Sending Touch (ACTION_UP): 0:(408.2985,775.55615)

:Sending Touch (ACTION_DOWN): 0:(268.0,617.0)

:Sending Touch (ACTION_UP): 0:(267.8827,603.579)

:Sending Touch (ACTION_DOWN): 0:(101.0,261.0)

:Sending Touch (ACTION_UP): 0:(98.0403,268.6024)

:Sending Touch (ACTION_DOWN): 0:(10.0,638.0)

:Sending Touch (ACTION_UP): 0:(17.424154,637.805)

:Sending Flip keyboardOpen=true

:Sending Touch (ACTION_DOWN): 0:(114.0,587.0)

:Sending Touch (ACTION_UP): 0:(144.14879,609.0429)

:Sending Touch (ACTION_DOWN): 0:(22.0,746.0)

:Sending Touch (ACTION_UP): 0:(21.255947,750.51385)

:Switch:
#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFla
gs=0x10200000;component=allclasses.emuwifimanager/.MainMenu;end

    // Allowing start of Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] cmp=allclasses.emuwifimanager/.MainMenu } in package
allclasses.emuwifimanager

:Sending Touch (ACTION_DOWN): 0:(446.0,683.0)

:Sending Touch (ACTION_UP): 0:(456.80484,755.7969)

:Sending Touch (ACTION_DOWN): 0:(165.0,378.0)

:Sending Touch (ACTION_UP): 0:(166.81113,376.14938)
```

```
:Sending Touch (ACTION_DOWN): 0:(112.0,392.0)
:Sending Touch (ACTION_UP): 0:(88.82703,446.68637)
:Sending Touch (ACTION_DOWN): 0:(329.0,729.0)
:Sending Touch (ACTION_UP): 0:(336.2307,721.04364)
:Sending Touch (ACTION_DOWN): 0:(133.0,169.0)
:Sending Touch (ACTION_UP): 0:(139.00114,159.12273)
   // Allowing start of Intent { cmp=allclasses.emuwifimanager/.NewLocationPointRetrieve } in
package allclasses.emuwifimanager
:Sending Touch (ACTION_DOWN): 0:(452.0,78.0)
   //[calendar_time:2014-05-18 08:35:39.895  system_uptime:51977758]
   // Sending event #300
:Sending Touch (ACTION_UP): 0:(480.0,0.0)
:Sending Touch (ACTION_DOWN): 0:(99.0,794.0)
:Sending Touch (ACTION_UP): 0:(107.54795,780.08954)
:Sending Touch (ACTION_DOWN): 0:(247.0,264.0)
:Sending Touch (ACTION_UP): 0:(238.40408,258.08496)
:Sending Touch (ACTION_DOWN): 0:(380.0,98.0)
:Sending Touch (ACTION_UP): 0:(387.91678,107.020325)
:Sending Touch (ACTION_DOWN): 0:(400.0,231.0)
:Sending Touch (ACTION_UP): 0:(384.0611,270.53253)
:Sending Touch (ACTION_DOWN): 0:(102.0,708.0)
:Sending Touch (ACTION_UP): 0:(98.68609,711.19934)
:Sending Trackball (ACTION_MOVE): 0:(2.0,-1.0)
:Sending Trackball (ACTION_MOVE): 0:(-3.0,3.0)
:Sending Trackball (ACTION_MOVE): 0:(-5.0,-3.0)
:Sending Trackball (ACTION_MOVE): 0:(3.0,-1.0)
:Sending Trackball (ACTION_UP): 0:(0.0,0.0)
   // Rejecting start of Intent { act=android.settings.LOCATION_SOURCE_SETTINGS
cmp=com.android.settings/.Settings$LocationSettingsActivity } in package com.android.settings
:Sending Touch (ACTION_DOWN): 0:(452.0,11.0)
:Sending Touch (ACTION_UP): 0:(480.0,74.81749)
```

```
:Sending Trackball (ACTION_MOVE): 0:(-1.0,-5.0)

:Sending Touch (ACTION_DOWN): 0:(365.0,455.0)

:Sending Touch (ACTION_UP): 0:(353.96414,440.09528)

    //[calendar_time:2014-05-18 08:35:40.172  system_uptime:51978035]

    // Sending event #400

    //[calendar_time:2014-05-18 08:35:40.173  system_uptime:51978035]

    // Sending event #400

    // Rejecting start of Intent { act=android.settings.LOCATION_SOURCE_SETTINGS
cmp=com.android.settings/.Settings$LocationSettingsActivity } in package com.android.settings

    // Rejecting start of Intent { act=android.settings.LOCATION_SOURCE_SETTINGS
cmp=com.android.settings/.Settings$LocationSettingsActivity } in package com.android.settings

:Sending Touch (ACTION_DOWN): 0:(252.0,407.0)

:Sending Touch (ACTION_UP): 0:(249.25243,401.62564)

:Sending Touch (ACTION_DOWN): 0:(162.0,700.0)

:Sending Touch (ACTION_UP): 0:(140.07066,726.95905)

:Sending Trackball (ACTION_MOVE): 0:(0.0,-2.0)

:Sending Touch (ACTION_DOWN): 0:(316.0,446.0)

:Sending Touch (ACTION_UP): 0:(330.36206,445.87384)

:Sending Touch (ACTION_DOWN): 0:(94.0,290.0)

:Sending Touch (ACTION_UP): 0:(102.93653,290.9221)

:Sending Trackball (ACTION_MOVE): 0:(-1.0,4.0)

:Sending Touch (ACTION_DOWN): 0:(262.0,294.0)

:Sending Touch (ACTION_UP): 0:(259.15402,292.1949)

    // Rejecting start of Intent { act=com.viber.voip.action.MESSAGES
cmp=com.viber.voip/.WelcomeActivity } in package com.viber.voip

:Sending Touch (ACTION_DOWN): 0:(188.0,602.0)

:Sending Touch (ACTION_UP): 0:(183.82263,563.1976)

:Sending Touch (ACTION_DOWN): 0:(44.0,625.0)

:Sending Touch (ACTION_UP): 0:(47.93326,601.4019)
:Switch:
#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFla
gs=0x10200000;component=allclasses.emuwifimanager/.MainMenu;end
    // Allowing start of Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] cmp=allclasses.emuwifimanager/.MainMenu } in package
```

```
allclasses.emuwifimanager

:Sending Touch (ACTION_DOWN): 0:(42.0,693.0)

:Sending Touch (ACTION_UP): 0:(41.823475,696.8878)

:Sending Touch (ACTION_DOWN): 0:(425.0,464.0)

:Sending Touch (ACTION_UP): 0:(374.63745,408.8032)

:Sending Touch (ACTION_DOWN): 0:(77.0,176.0)

:Sending Touch (ACTION_UP): 0:(90.53031,143.42264)

:Sending Trackball (ACTION_MOVE): 0:(-2.0,1.0)

:Sending Touch (ACTION_DOWN): 0:(279.0,172.0)

:Sending Touch (ACTION_UP): 0:(288.57446,158.92351)

Events injected: 500

:Sending rotation degree=0, persist=false

:Dropped: keys=1 pointers=0 trackballs=0 flips=0 rotations=0

## Network stats: elapsed time=6434ms (0ms mobile, 6434ms wifi, 0ms not connected)

// Monkey finished
```