Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 12, December -2017

Heterogeneous Network Management Functionalities For Better Communication Quality

Charu Gupta[1]

ME-Digital Communication, Department of Electronics & Communication Engineering, MBM Engineering College, JNV University, Jodhpur-342011

ABSTACT- *This paper provides historical overview of operating systems for mobile devices which builds the foundation for the usage of the Android mobile platform. This introduction is not connected to any product or brand and is therefore usable for everyone. The paper compares Android to the already existing and mostly used platforms "Symbian OS" and "Windows Mobile". This deepens the understanding and shows the main advantages and disadvantages. The paper anticipates with application of android for Telematics taking in account android system design to utilize multiple network access and Telematics services.*

Keywords *-Telematics, Heterogeneous network, Android.*

I. INTRODUCTION

Before Android, mobile developers faced many roadblocks when it came to writing applications. Building the better application, the unique application, the competing application, the hybrid application, and incorporating many common tasks such as messaging and calling in a familiar way were often unrealistic goals. The Motorola Dyna TAC 8000X was the first commercially available cell phone. First marketed in 1983, it was 13 x 1.75 x 3.5 inches in dimension, weighed about 2.5 pounds, and allowed us to talk for a little more than half an hour. We called it "The Brick," and the nickname stuck for many of those early mobile phones we alternatively loved and hated. About the size of a brick, with a battery power just long enough for half a conversation, these early mobile handsets were mostly seen in the hands of traveling business execs, security personnel, and the wealthy. First-generation mobile phones were just too expensive. The service charges alone would bankrupt the average person, especially when roaming. Early mobile phones were not particularly full featured. These early phones did little more than make and receive calls and, if you were lucky, there was a simple contacts application that wasn't impossible to use. These first-generation mobile phones were designed and developed by the handset manufacturers. Competition was fierce and trade secrets were closely guarded.

Manufacturers didn't want to expose the internal workings of their handsets, so they usually developed the phone software in-house. These early phones were flawed, but they did something important—they changed the way people thought about communication. As mobile phone prices dropped, batteries improved, and reception areas grew, more and more people began carrying these handy devices. Soon mobile phones were more than just a novelty. Customers began pushing for more features and more games. But, there was a problem. The handset manufacturers didn't have the motivation or the resources to build every application users wanted. They needed some way to provide a portal for entertainment and information services without allowing direct access to the handset.By this time, professional Web sites were full color and chock full of text, images, and other sorts of media. These sites relied on JavaScript, Flash, and other technologies to enhance the user experience

* P. G. Scholar, Department of Electronics & Communication Engineering, Fac. of Engg. & Arch., M. B. m. E. C., J.N.V. University, Jodhpur

and were often designed with a target resolution of 800x600 pixels and higher. Mobile devices have steadily gained acceptance as a multimedia platform. Current tools offer application developers options to use various technologies—for example, Java, Open C, Python, Flash Lite, XHTML/CSS, JavaScript, and Mobile Ajax—to implement highly functional mobile applications. Content developers can work with audio, video, multimedia messaging, and Flash to create rich and compelling mobile content. Although the choice of development platform is largely market-driven, it also depends on the characteristics of available platforms and the requirements of particular applications.

Handset manufacturers realized that if they wanted to continue to sell traditional handsets, they needed to change their protectionist policies pertaining to handset design and expose their internal frameworks, at least, to some extent. A variety of different proprietary platforms emerged—and developers are still actively creating applications for them. Some Smartphone devices ran Palm OS (now Garnet OS) and RIM Blackberry OS. Sun Microsystems took its popular Java platform and J2ME emerged (now known as Java Micro Edition [Java ME]). Chipset maker Qualcomm developed and licensed its Binary Runtime Environment for Wireless (BREW). Other platforms, such as Symbian OS, were developed by handset manufacturers such as Nokia, Sony Ericsson, Motorola, and Samsung. The Apple I Phone OS (OS X I Phone) joined the ranks.

For manufacturers and mobile operators, handset product lines became complicated fast. Platform market penetration varies greatly by region and user demographic. Instead of choosing just one platform, manufacturers and operators have been forced to sell phones for all the different platforms to compete. For instance, Symbian phones often also support J2ME.The mobile developer community has become as fragmented as the market. It's nearly impossible to keep track of all the changes in the market. Developer specialty niches have formed. The platform development requirements vary greatly. Mobile software developers work with distinctly different programming environments, different tools, and different programming languages. Porting among the platforms is often costly and not straightforward. Keeping track of handset configurations and testing requirements, signing and certification programs, carrier relationships, and application marketplaces have become complex spin-off businesses of their own.

Android, Inc. was founded in Palo Alto, California, United States in October, 2003 by Andy Rubin, Rich Miner, et al. to develop, in Rubin's words "...smarter mobile devices that are more aware of its owner's location and preferences." Key employees involved in the founding of Android Inc. include Andy Rubin, also the co-founder of Danger Inc., Andy McFadden, who worked with Rubin at WebTV, and Chris White, who led the design and interface of WebTV Other crucial employees includes Richard Miner, a co-founder of Wildfire Communications, Inc. and former vice-president of Technology and innovation at Orange, and all those of whom brought considerable wireless industry experience to the company. Despite the obvious past accomplishments of the founders and early employees, Android Inc. operated secretively, admitting only that it was working on software for mobile phones. The Android mascot is a little green robot as shown in the figure

Google acquired Android Inc. in August, 2005, making Android Inc. a wholly-owned subsidiary of Google Inc. Key employees of Android Inc., including Andy Rubin, Rich Miner and Chris White,
stayed at the company after the acquisition. Google's motivation for supporting the Android project

Figure 1 Android Mascot

seems to be having Android everywhere and by doing that, creating a level playing field for mobile devices. Ultimately, Google is a media company, and its business model is based on selling advertising. If everyone is using Android, then Google can provide additional services on top of it and compete fairly. This is unlike the business models of other software vendors who depend on licensing fees. Although Google does license some proprietary apps, such as Gmail and Maps, and makes some money off the Android market, its primary motivation is still the advertising revenue that those apps bring in.
The Android operating system was designed from the ground up to be a comprehensive open source platform for mobile devices. It is a game-changer in the industry and it is enjoying great success.

II. GOOGLE ANDROID

The term "Android" has its origin in the Greek word *andr-*, meaning "*man* or *male*" and the suffix - *eides*, used to mean "*alike* or *of the species*". This together means as much as "being human". Android is a software stack for mobile devices which means a reference to a set of system programs or a set of application programs that form a complete system. This software platform provides a foundation for applications just like a real working platform.

Android is a comprehensive platform, which means it is a complete software stack for a mobile device. For developers, Android provides all the tools and frameworks for developing mobile apps quickly and easily. The Android SDK is all you need to start developing for Android; you don't even need a physical phone. For developers, Android provides all the tools and frameworks for developing mobile apps quickly and easily. The Android SDK is all you need to start developing for Android; you don't even need a physical phone. Android is an open source platform. The entire stack, from low-level Linux modules all the way to native libraries, and from the application framework to complete

applications, is totally open. More so, Android is licensed under business-friendly licenses (Apache/MIT) so that others can freely extend it and use it for variety of purposes.
Google Android is a recent operating system, designed for mobile devices that perfectly fits to embedded devices such as those used for automotive infotainment. A fundamental feature of Android consists in its openness: a free SDK (Software Development Kit) is available for developers who wish to address this platform. Android is a software platform developed specifically for mobile devices, it includes an operating system, middleware and a few bundled applications. The full project is released under Apache version two license, which basically has no copy left clause. In agreement with the Web 2.0 paradigm, information sharing between different processes and applications is possible though content providers. The platform provides a few native content providers (e.g. media, contacts, etc.) and new ones can be added, so enabling the developers to build rich peer-to-peer applications.

2.1 Architecture overview:
The high-level architecture of Android consists of five main components:

- Linux kernel,
- Libraries,
- Android runtime,
- Application framework,
- Applications.

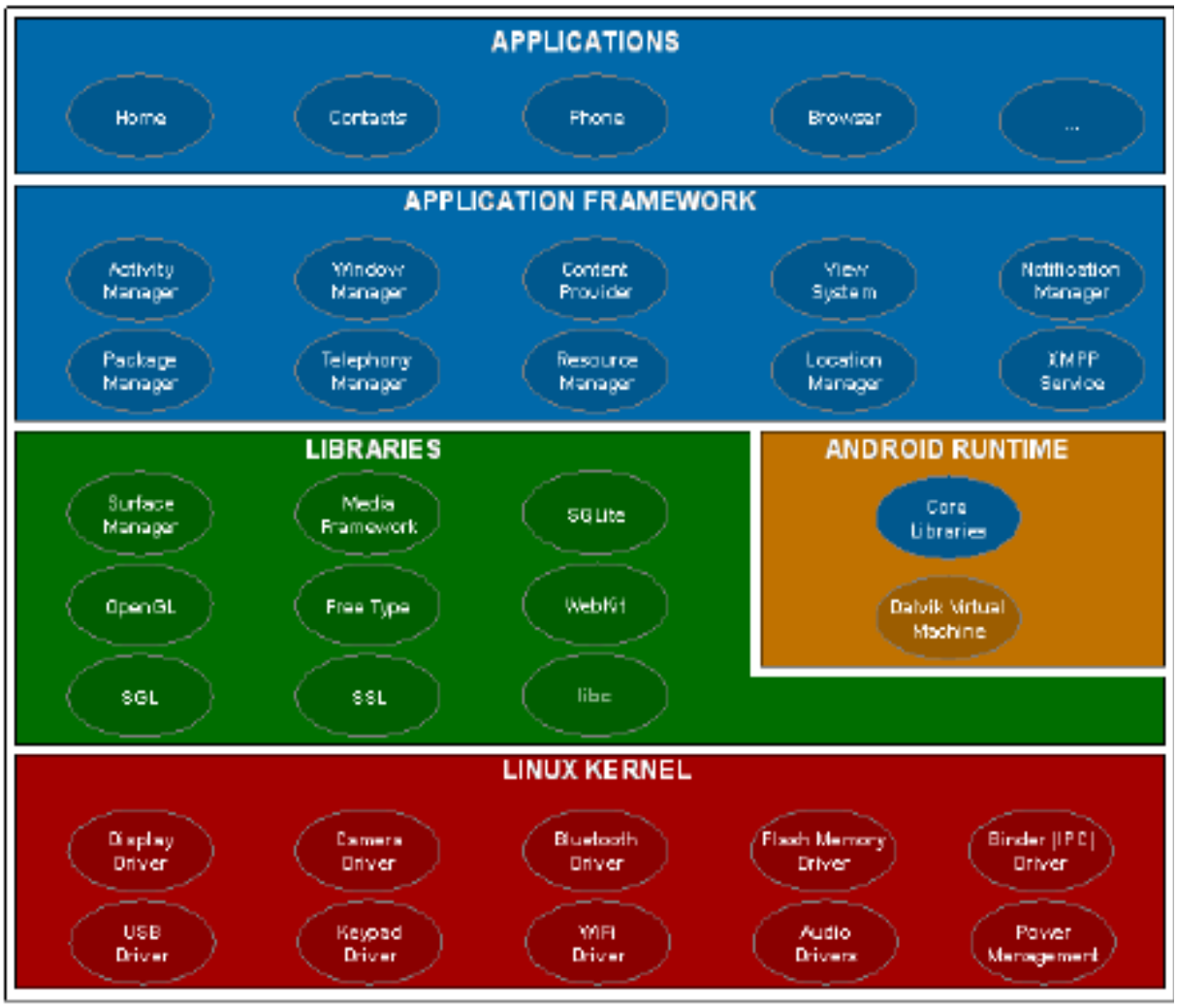

Figure 2. Android Architecture

At the bottom of the hierarchy there is the Linux Kernel. Basically, it is the 2.6.27 version of Linux Kernel, to which are applied Android specific patches. This element is responsible of managing the core system services and driver model. The root files system uses rootfs, whereas data and system are using YAFFS, which is a file system specifically designed for NAND and NOR Flash drives. The application framework and Android runtime rely on a set of C/C++ libraries. The set of libraries include the standard C libraries, media libraries, graphical libraries, a browser engine (LibWebCore), font libraries (Free Type), and database libraries (SQLite). The Android runtime consists of Core libraries and The Dalvik Java virtual machine. Dalvik is optimized to allow multiple instances of the virtual machine to run at the same time using

a limited amount of memory. Each instance runs in a separate Linux process. The application framework is a large set of classes' interfaces, and packages. Its goal is to provide an easy and consistent way to manage graphical user interfaces, access resources and content, receive notifications, or to handle incoming calls. The main components are: the view system, the activity manager, content providers, the resource manager, the notification manager, and the telephony manager.

2.2 Inter-application communication:

Google Android has two inter-application communication modes: *intents* and *code binding*. The *intents framework* provides a high level Inter Process Communication (IPC). This is the best way to implement dynamic functionality binding between applications developed using the SDK. The Intent class contains several fields describing what a caller would like to do. Intent fields include the desired action, category, and data string, MIME type of the data, handling class, and security restrictions. Intents can be used to launch activities, to send data in broadcast, and to start services. The security restrictions are implemented using the *permissions framework* provided by Android.

In the Android platform one process normally cannot access the memory of another process. Therefore to communicate, two processes need to decompose their objects into primitives that the operating system can understand, and marshal the object across the process boundary. The Android Interface Definition Language (AIDL) tool provided with SDK creates the marshalling code automatically. AIDL is an Interface Description Language (IDL) used to generate code that enables two processes to interact using IPC. The AIDL IPC mechanism uses a proxy class to pass values between the client and the implementation.

Basic application components of android have been listed in the table below:

Activities	UI component typically corresponding to one screen.
BroadcastReceivers	Respond to broadcast Intents.
Services	Faceless tasks that run in the background.
ContentProviders	Enable applications to share data.

Figure 3. Application Component of Android

Activities: Typically correspond to one screen in a UI. But, they can:

- be faceless
- be in a floating window
- return a value

Intents: Think of Intents as a verb and object; a description of what you want done Examples: VIEW, CALL, PLAY, etc. System matches Intent with Activity that can best provide that service. Activities and Broadcast Receivers describe what Intents they can service in their Intent Filters (via AndroidManifest.xml)

Figure 4. Intents

Broadcast Receivers: Components designed to respond to broadcast Intents. Think of them as a way to respond to extern notifications or alarm Applications can invent and broadcast their own Intents as well

Services: Faceless components that run in the background Example: music player, network download, etc. Bind your code to a running service via a remote-able interface defined in an IDL Can run in your own process or separate process

Content Providers: Enables sharing of data across applications Examples: address book, photo gallery, etc. Provides uniform APIs for: querying (returns a Cursor) delete, update, and insert row. Content is represented by URI and MIME type.

2.3 Important features integrated in Android:

Android offers many features cover many areas such as application development, internet, media and connectivity. Some of the most important ones are presented in the following list:

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source Web Kit engine
- Optimized **graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media **support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and Wi-Fi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich **development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE.

III. COMPARISION OF ANDROID WITH OTHER OS

A mobile system is a computer system which isn't linked to a certain place. It is possible to move it or carry it around like e.g. a cell phone, a handheld or a special computer system in a car. Mobile devices have changed their profile dramatically in the last years. The advanced mobile phones of today integrate fully-featured personal digital assistant (PDA) capabilities with those of a traditional mobile phone. Here we examine the critical factors for operating systems in this market which differentiate them from each other. The classification of operating systems has to consider the market in which they are used. The market for advanced mobile devices is hard to compare to other markets like the PC market where also operating systems are used. User needs and requirements are different.

For the purpose of comparison. Technical aspects of these systems have to be considered; also user needs are very important. Because user needs differ the identification of an ideal operating system is not possible. Only a classification or an optimal solution relating to a certain group of individuals is possible. In the following we will have a look at classification criteria which are important to compare operating systems.

Issue description	Java ME	.NET CF	Flash Lite	Android
Software architecture and technical issues				
Footprint	~128 Kbytes for storage of kernel-based virtual machine and associated libraries	1.55 Mbytes on Windows Mobile-based Pocket PC 2000/2002; 1.35 Mbytes on Windows Mobile for Pocket PC 2003 or Windows CE .NET devices	450 Kbytes for the core library of Flash Lite 2.1; 374 Kbytes for Flash Lite 3.1	3 Mbytes
Runtime memory requirement	< 0.5 Mbytes	~ 0.5 Mbytes	2–4 Mbytes	Minimum 32 Mbytes of RAM
Memory management	Automatic memory management provided by the traditional garbage collector, which deallocates memory occupied by objects that the program no longer uses	Automatic memory management provided by Common Language Runtime (CLR); the CLR garbage collector manages the allocation and release of memory for an application	Garbage collection executed automatically every minute or whenever an application's memory use increases by 20 percent or more	Automatic memory management handled by Dalvik's garbage collector; garbage collections might noticeably decrease performance
Device support	All devices support Connected Limited Device Configuration (CLDC), Mobile Information Dance Profile (MIDP) (practically, lacks support only for Windows Mobile-based Pocket PCs)	Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003-based Pocket PCs and smartphones, embedded systems running Windows CE .NET 4.1 and later	Mobile phones and PDAs from major manufacturers such as Fujitsu, Hitachi, LG, Mitsubishi, Motorola, Nokia, Panasonic, Samsung, Sanyo, Sharp, and Sony Ericsson	Mostly HTC devices (Magic, Hero, Tattoo); also T-Mobile (G1, Pulse), Motorola Dext, Samsung Galaxy i7500, Acer Liquid, Sony Ericsson Xperia X10; Android 2.0-compatible handsets announced (Motorola, Samsung)
User interface (UI) components	High-level LCDUI components, such as Form or List; low-level LCDUI for controlling every UI pixel; support for SVG (defined in JSR 287); J2ME Polish allows design along with animations and effects specified in external CSS-like files	Windows Forms controls (vary for Pocket PCs and smartphones)	Nokia Flash Lite Feather Framework (FL 2.x), Sony Ericsson Adobe XD UI components (FL 1.1/2.x)	View and ViewGroup objects; DroidDraw tool serves for rapid UI design; J2ME Polish enables conversion of Java ME MIDlets' UI to Android-compatible UI
Development languages	Java (CLDC/MIDP)	C#, Visual Basic .NET	ActionScript 1.0, ActionScript 2.0	Java (Android SDK)
Packaging	Java Application Description (JAD) and Java archive (JAR) files	Cabinet (CAB) file installers	SWF files	Android package (APK) files
Deployment methods	Over the air (OTA), Bluetooth/IR, Wireless Application Protocol (WAP) push	OTA, Bluetooth	Bluetooth, physical cable, OTA	OTA, Bluetooth
Server-side technologies*	Java servlets, JavaServer Pages (JSP)	ASP.NET Mobile Controls	Flash Media Server (uses ActionScript 1 for server-side logic)	Java servlets, JSP
Persistent storage and database support	RMS and Perst Lite from mObject	Local database support for SQL Server Mobile Edition; on the server side, support for SQL Server	Persistent storage through shared objects; on the server side, support for interaction with PHP scripts and use of MySQL database	Android APIs contain support for SQLite database

Figure 5. Comparison of Android with other OS

The proposed heterogeneous network management algorithm integrates the channel estimation information to estimate the change of adaptive signal-tracking decay, and also improves the estimation handover correctness and reduces the system computation load.

IV. ANDROID SYSTEM DESIGN AND IMPLEMENTATION FOR TELEMATICS SERVICES

The rapid progress of wireless communication in recent years extends the information and communication services from digital homes to mobility life. Different types of wireless communication technologies are widely applied to our daily life so that the demands for accessing internet resources through wireless devices also increase rapidly.

This application aims to design an Android-based mobile device platform that integrated with network management functions and MOST (Media Oriented Systems Transport) technologies to utilize multiple network access and Telematics

services. The platform composed a heterogeneous network management algorithm which includes roaming and sharing functions to maintain the qualities of services and satisfy different kinds of scenario needs. The proposed heterogeneous network management algorithm not only manages the heterogeneous networks handover, but also provides efficient resource sharing in the heterogeneous network environment. We also design the seamless handover functionality in the proposed algorithm to support the ubiquitous computing. Implementation of the proposed heterogeneous network management algorithm is based on the Android-based mobile device platform. The proposed Android-based mobile device platform integrating with heterogeneous network technologies can be applied for many scenarios in our life to realize ubiquitous services sharing and roaming for various network environments.

4.1 System Architecture:

The developed system integrates technologies of heterogeneous networks management and Android software system design. The system can serve as a Home Entertainment Gateway, a Vehicle On-Board Unit (OBU), or a portable network device. People can access the wired/wireless Internets more conveniently and effectively via the system. The developed system equipped with heterogeneous network interfaces can provide with two functions: Roaming and Sharing. The Roaming function indicates that the system can search and choose the wireless network with best resources (ex: RSS, Bandwidth) for users according to the surrounding environment. This is especially helpful when the system is used for roaming between the homogeneous or heterogeneous networks. The Sharing function indicates that the system can allocate the available resources to all users when the system is connected with multiple wired/wireless networks.

The heterogeneous networks management technology is very important to be implemented for analyzing the multiple network/user conditions, and deciding the resource allocation policy. The communication link of network and physical layer are adjusted adaptively based on the handover decision. The technology also considers the seamless roaming issues to provide better QoS (quality of service) for users.

4.2. Algorithm Design:

4.2.1 Methodology

A heterogeneous network management algorithm has been designed to maintain the qualities of access networks and services. The proposed algorithm shown in Fig. cooperates with Roaming function and sharing function according to the situational application. When the system starts up, the system will detect the available network resources and the status of environment. The system decides to use Roaming or sharing function according to this information or by user commands. The algorithm of the heterogeneous network management can be divided into three processes, and the details of each process are described as follows:

1. Handover Control Process:

The middle part of the flowchart shows the process of handover control which includes four steps from physical layer to application layer to guarantee the QoS. The first step of this process is handover initialization step. The Network Analyzer function monitors the status of networks and predicts the power gain or SINR (Signal to Interference and Noise Ratio) variation of each network. When the system discovers the signal qualities of networks occurring significant variation, the Network Analyzer function notifies the Roaming Function and Sharing Function Processes for the handover initialization.

At the second step of this process, the On-line Service Analyzer function receives the information of available network resources sent from Search New Link module of the Roaming Function Process and examines the characteristics of user services sent from the user/service management module of the Sharing Function Process. Then it decides whether to execute vertical or horizontal handover and reallocates resources to on-line services based on their priorities.

At the third step of this process, if the handover type of a service is vertical handover, the Higher Layer Establish function will setup a new service thread for this service. The new service thread uses the primitives which are established by the vertical handover and Network Link Establish modules to communicate with the new network attachment point (NAP) and continues the old service process. The new service thread will connect to the other communication party via the primitives which establishes new network layer and link layer connections using Heterogeneous NIC. After the new service thread is established, the old service thread connected to the original network attachment point breaks. This is intended to provide seamless services.

At the finally step, the system will release the old service process and the old network resource to complete the handover control process. The services will be continued by applying the handover control process at upper layer even if the physical communication link is changed when roaming between the heterogeneous networks.

2. Roaming Function Process

The right half part of the flowchart shows the process of the Roaming Function which manages the physical layer handover of the system. The Roaming Function is responsible for choosing the most appropriate network interface according to the surrounding environment. When the Roaming Function is enabled, the system analyzes the qualities of used networks. The Search New Link function checks the network information sent from the Network Analyzer function to see if necessary to search the new network link for the handover preparation or not. When the quality of used network

is going down, the system will search new candidates with better link performances. Next, the Network Layer Establish function will reserve the old IP and routing table information to keep the connections of used services continuously on original communication links, until new IP and routing table of new communication link has been setup. And then applies the Higher Layer Establish function of Handover Control Process to execute the seamless service handover at Application Framework Layer. If there is only single wireless network interface, the system will execute the horizontal handover to make the network interface connecting to the NAP with better signal strength or bandwidth according to the information sent from Search New Link function.
Next, the Network Layer Establish function will change the IP and routing table information according to the new NAP for the change of communication links, and then applies the Services Handover function of Handover Control Process to execute the service handover at Application Layer. The handover functions will send the selected NICs' information including RSSI, SINR, and QoS parameters to the Network Analyzer function of Handover Control Process for analyzing the qualities of used networks and services handover constraints.

3. Sharing Function Process

The left half part of the flowchart shows the process of the Sharing Function which uses Resource Management function to control the network layer handover of the system. The Sharing Function is used to allocate the available network resources for users. When the Sharing Function is enabled, the system analyzes the qualities of used networks. The User Link Detection function detects the user connections and the active NIC statuses sent from Network Analyzer function to see if any network resource varied. If there exists any NIC being removed from the system or added to it, the Network Capability Setup function (NCS) will analyzes the QoS parameters sent from the User Link Detection function, and classifies the connected networks into several groups of different qualities. If there exist any user joining to the sharing network or leaving from it, the Service Requirement Setup (SRS) function will update the users' information which includes user preference and the characteristics of online services (ex: the bandwidth, service type, delay, packet loss rate, etc.). Then SRS classifies services into several groups of different QoS requirement according to the service characteristics.
As either of NCS and SRS is enabled, the Resource Management function will be triggered to rearrange connections between the NICs and services according to the information provided by NCS function and SRS function. The loading ratio of each service group of each NIC will be derived. Then the service number in each NIC could be decided. Finally, the Network Layer Setup function setups the network routing tables with user services and NIC configured by Resource Management function. The results of the resource management will be sent to the Service Handover function to execute the Handover Control Process.

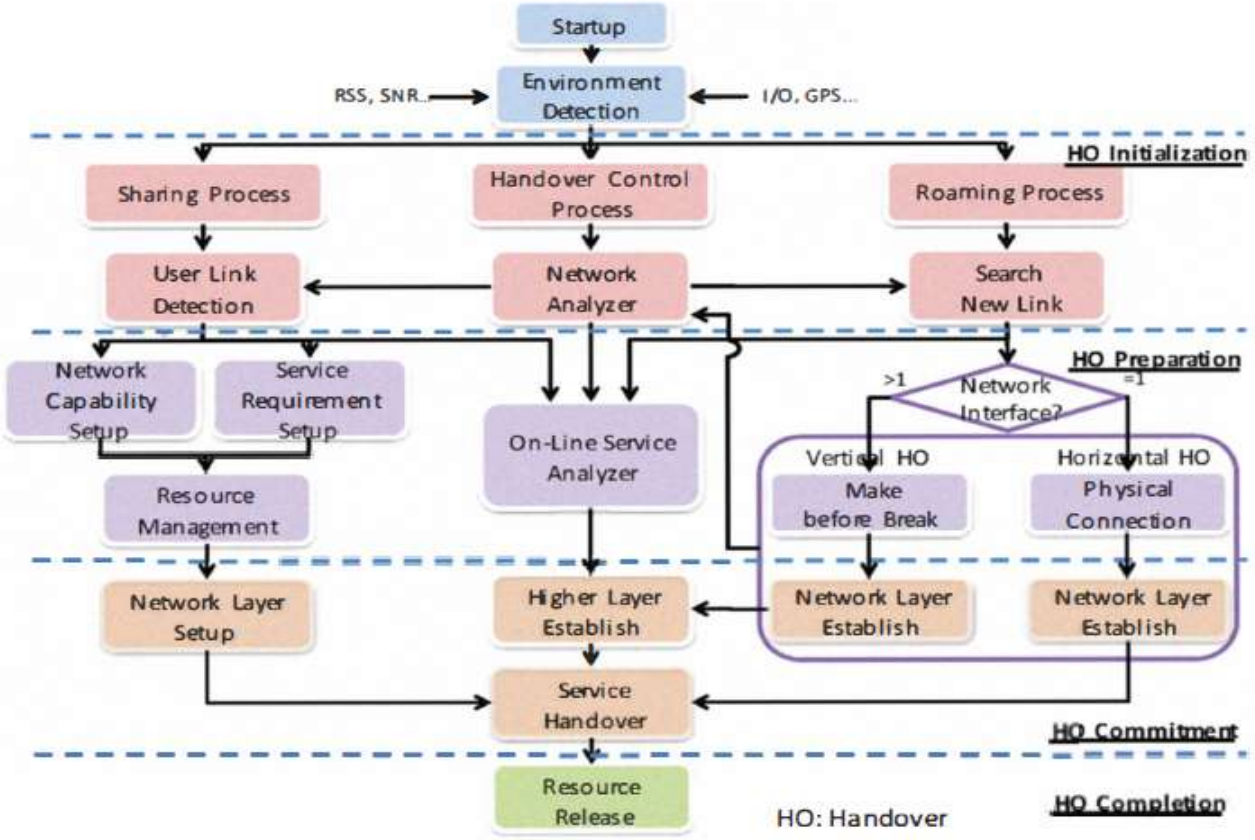

Figure 6. Flow Chart of Heterogeneous Network

V. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed an Android-based mobile device platform which provides the heterogeneous network management functionalities to guarantee the communication quality. In order to achieve ubiquitous computing, the proposed algorithm supports seamless handover via effective and rapid resource and handover managements between heterogeneous networks. In the future, we will design a more sophisticated resource management algorithm to enhance the QoS during the network roaming and sharing processes. The algorithm will provide all resources of heterogeneous network interfaces of the system for multi-users sharing, and achieve an intelligent load-balance state of the system.

REFERENCES

(a) "*Development Platforms for Mobile Applications: Status and Trends*", Damianos Gavalas and Daphne Economou, University of the Aegean. (2011)
(b) "*An Android System Design and Implementation for Telematics Services*", Yong-Hua Cheng, Wen-Kuang Kuo, Szu-Lin Su, Institute of Computer and Communication Engineering Department of Electrical Engineering National Cheng Kung University, Tainan, Taiwan.(2010)
(c) "*Android/OSGi-based Vehicular Network Management System*", Teng-WenChang *Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan*.(2010).
(d) Kim, C.S., Kim, J.I., Han, W.Y., Kwon, O.C., "Development of OpenTelematics Service Based on Gateway and Framework", Proc. of theICACT, 2006, pp.1349-1352.
(e) Han, W.Y., Kwon, O.C., Park, J.H., Kang, J.H., "A Gateway and Framework for Interoperable Telematics Systems Independent on Mobile Networks", ETRI Journal, Vo1.27, No.1, 2005, pp.106-109.
(f) Gomi, Y., Weiland, RJ., "An Open Platform for Telematics", Proc. of the ITS World Congress, 2005.
(g) Kim, C.S. Kim, J.I., Kwon, O.C., "Telematics Transport Gateway for Telematics Systems Independent on Mobile Networks", Proc. of the ITS World Congress, 2005.

