SMARTPHONES AND ANDROID INTERNALS

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Ravish H. Thakor
Fall 2012
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Ravish H. Thakor:

Smartphones and Android Internals

______________________
N. Marovac, Chair
Department of Computer Science

______________________
Roman Swiniarski
Department of Computer Science

______________________
Lawrence Rhyme
Management Department

11/09/2012
Approval Date
Copyright © 2012
by
Ravish H. Thakor
All Rights Reserved
DEDICATION

This thesis could not have been a success without the constant support and guidance of my thesis advisors, Dr. Nenad Marovac, Dr. Roman Swiniarski and Dr. Lawrence Rhyne. Dr. Nenad Marovac offered me continuous encouragement and guidance. He suggested the work flow for the thesis. He was constantly encouraging me and provided me with his valuable time to discuss the contents also. Dr. Roman Swiniarski was always there, when I needed support for the thesis. Dr. Lawrence Rhyne was also a great help and resource during this thesis documentation, providing me with constant guidance during the writing of the thesis draft. All three of my thesis advisors have been a constant help during this thesis work. Therefore, I would like to express my deepest gratitude to all three of my thesis advisors.
ABSTRACT OF THE THESIS

Smartphones and Android Internals
by
Ravish H. Thakor
Master of Science in Computer Science
San Diego State University, 2012

This thesis is focused on a detailed study of smart phones with Android operating system in particular. It is focused on Android, covering its history, basics, various features, architecture etc. Not only professionals but every tech. literate person are using smart phones and rely on them for their daily tasks. Android operating system has created a benchmark for smart phone operating systems. This is clearly depicted in recent surveys on market domination among smart phone Operating Systems. Android applications are cataloged for general use on an online market place and can be downloaded via a web connection by owners of Android Smartphone. The numbers of applications are increasing rapidly. The main reason behind its success is that Android is open-sourced and its programming is highly leveraged from Java.

The research work has been presented in an easy to understand and lucid manner and requires no pre-requisite knowledge about Android OS. This thesis attempts to introduce new Android developers to different Android flavors, its features and Android architecture. It also provide detailed assistance in certain areas such as Android boot up process, radio interface layer, Attention Commands, logging and debugging mechanism, and upgrade for the Android devices. At this time, there is a paucity of texts to perform these functions.

This thesis is very useful, for those who are not familiar to Android. It introduces Android as a benchmark among other mobile operating systems, Android versions, and the different software layers interacting within the device. Nowadays each smartphone has an upgrade ability. We discuss different types of upgrades and their working mechanisms. We have tried to provide further details informing the consumers as well as developers about the process of analyzing an issue or a flaw detected during test procedures and how logging and debugging methods related to android would be helpful in finding the root cause of the error.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction to Smartphone</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Software Operating System Platforms</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Why Android</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Research Objective</td>
<td>5</td>
</tr>
<tr>
<td>2 INTRODUCTION TO ANDROID</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Introducing Android</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Features Integrated in Android</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Android Platform Versions</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Current Distribution of Android Version</td>
<td>11</td>
</tr>
<tr>
<td>2.5 Android Architecture</td>
<td>11</td>
</tr>
<tr>
<td>3 ANDROID INTERNALS</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Boot Up Process in Android</td>
<td>14</td>
</tr>
<tr>
<td>3.1.1 Stage 1: The Android Boot Process from Power On</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2 Stage 2: The Boot Loader</td>
<td>15</td>
</tr>
<tr>
<td>3.1.3 Stage 3: The Linux Kernel</td>
<td>16</td>
</tr>
<tr>
<td>3.1.4 Stage 4: The Init Process</td>
<td>16</td>
</tr>
<tr>
<td>3.1.5 Stage 5: Zygote and Dalvik</td>
<td>16</td>
</tr>
<tr>
<td>3.1.6 Stage 6: The System Server</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Introduction to the Attention Commands (AT Commands)</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Radio Interface Layer in Android</td>
<td>19</td>
</tr>
<tr>
<td>4 THE LOGGING AND DEBUGGING MECHANISM</td>
<td>22</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>22</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>World-Wide Smartphone Sales</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Current Distribution of Different Android Versions</td>
<td>11</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1. The first hand held cell phone ................................................................. 2
Figure 2.1. Android architecture .............................................................................. 12
Figure 3.1. Initial power up .................................................................................... 14
Figure 3.2. Boot loader ........................................................................................... 15
Figure 3.3. System server ....................................................................................... 17
Figure 3.4. The Android telephony architecture ..................................................... 20
Figure 4.1. Working flow of ADB .......................................................................... 24
Figure 4.2. Showing how to capture the screen using DDMS .................................. 26
Figure 4.3. Showing Logcat process in DDMS ....................................................... 27
Figure 5.1. Generating and deploying a DELTA file to update a handset with FOTA 34
Figure 5.2. Showing package on device ................................................................. 35
Figure 5.3. Package is downloading in the background ............................................ 36
Figure 5.4. Package is ready to update the device .................................................. 36
Figure 5.5. Software update is start to run ............................................................. 37
CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO SMARTPHONE

Communication between parties geographically distant from each other, whether it is a mile or 1000 miles, has become extremely simple and inexpensive. From the time when Alexander Graham Bell invented the first phone in 1876, this medium of communication has grown by leaps and bounds [1]. Stationary units were replaced by wireless ones with huge form factors and limited capabilities. These huge devices were replaced by smaller ones and some more added features like calculators, alarm clocks, etc. Now phones no longer adhere to the old definition. Phones, nowadays, are small computers running a plethora of gadgets with diverse functionality and high end performance. They are now called smartphones.

Since the first mobile phone, cellular phones and the technology involved with them have changed remarkably. Features like watches, calendars, alarm clocks, etc., are now integrated into them, so that the need to have separate entities to perform these functions has been eliminated. Cell phones are now more than mere a media for communication. Figure 1.1 [2] is showing the first hand held phone which is called DynaTAC.

On April 3, 1973, key member of Motorola, Dr. Martin Cooper, and his team unveiled the first hand held mobile phone for use on a cellular network [3]. It was named DynaTAC and it can be called the stepping stone for all major breakthroughs in mobile industry. Subsequently, different features and functions were added to the cell phone starting with first generation 1G and progressing on to the second generation 2G, to the third generation 3G. Fourth generation 4G, is the current and the most advanced technology for today telecom industry.

First generation was used only for voice communication. Real functions like calling (voice and data) and Short Message Service (SMS) were incorporated in the Second generation technology. Streaming applications were introduced in the Third generation technology [4]. Fourth generation has all the features and functions from all three generations with better performance, speed and high throughput for mobile technology [4]. The most
important advancement in the mobile phones functionality is the integration of the internet. Mobile phones made internet portable and easily accessible from a hand-held device.

In general, technology and computers specifically have long been moving towards mobility[5]. Not only does the form factor of the cellular phone lend itself to the ultimate portability, and pocket ability, but the move toward the consolidation of multiple features in the modern cell phone increasingly enables you to use the phone for tasks that you once needed to carry a separate device to accomplish [5]. The most important feature of a smartphone, aside from other features discusses further, is its ability to constantly connect to the Internet, which is truly revolutionary [4].

A Smartphone is essentially a phone which has almost all capabilities and functionality of a portable personal computer. Its predecessor, which is also called a feature
phone (in layman’s terms), has limited functionality compared to a smartphone, offering basic functions like voice call, SMS (short message service), MMS (multimedia message service), etc. A Smartphone, on the other end, contains certain added functions like web browsing, access to social networking websites like Face-book and Twitter, Location-based services, a high quality camera, Instant messaging services, Qwerty keyboard, various media file support, mailing services, games, and application support.

Smartphones have different hardware and software components which are essential to their overall structure. Hardware components include a modem, an antenna and an I/O interface as in a desktop or a laptop. Software components basically include an operating system (OS) and an application extension related framework associated with it. Operating system (OS) is one of the most important components in a smartphone. There are many mobile operating systems (OS) currently in use, each having different functions and features.

**1.2 SOFTWARE OPERATING SYSTEM PLATFORMS**

The list below mentions the most commonly and extensively used mobile operating systems.

- **IOS by Apple Inc.:** Since the release of Apple I-phone powering on the IOS operating system, it has become the most coveted product in the industry. It set the standards for smartphone. It provides internet and internet based applications in the device. It also create platform for developers to extract a rent. Apple is only making I phone as a smartphone [6].

- **Windows Operating System (OS) for Windows Mobile:** Nokia, HTC, Samsung and so many other device manufacturing companies are making Windows based smartphone. Windows Mobile has put all aesthetics of the desktop version of Windows to the smartphone, with similar behavior and same characteristic. The applications are using standard Microsoft tools and languages, like Visual Studio, .Net and Microsoft Office, etc. It has also market place for applications and games [5].

- **Blackberry Operating System (OS) from Research in Motion (RIM):** Research in Motion company develop the blackberry. It is the best suitable for the business users who like the instant push email services through the Blackberry server. It is programmed in C and it is also a closed source project. Development on the Blackberry platform revolves around the use of Java and the Blackberry Java Development Environment (JDE) [5].

- **Palm/HP:** Palm WebOS supports smartphones like PDAs. It needs to convert its operating system for smartphone, so WebOS was introduced. It was introduced in 2009. But, it fails to make a good market because of Android and Apple competition.
Later on HP purchased Palm and acquired WebOS and Palm’s many patents. Now main strength of WebOS is its sponsor HP [6].

- Android: In 2008, Google launched its first attempt at a smart phone operating system. Android is an open source initiative. It is mainly programmed in C, C++, and JAVA. It is an open source which can be used by any company that wishes to manufacture a hand held device with certain restrictions and regulatory measures to control the quality of the product. Google Inc. is involved with a group of manufacturers called the Open Handset Alliance which includes device manufacturers like HTC, Samsung, LG, Motorola, and mobile Carriers like AT&T, T-Mobile, Sprint Nextel, Semi conductor companies like Qualcomm, Intel, Texas Instrument, NVIDIA, Software companies like Google, EBay, and commercialization companies like Accenture, Wipro etc. [7, 8].

1.3 WHY ANDROID

Above we provided a description of different operating systems (OS) but among all of them, Android is unique for the qualities listed above. We are about to analyze the reasons why Android is so popular, and why I selected Android as an important part of this thesis:

1. Android is an open source: Developers have a big advantage due to open source. They can easily add/update the applications without any issues.

2. Android is powered by Google: Google acquired Android in 2005 and released the first phone in 2008. Google is a service provider for tools like blogging, maps, news, books, photo sharing, finance, shopping, email, apps, YouTube and Google Voice. Google earns no revenue from any device manufactures for services, operating system or basic apps. Google generates its revenue mainly from search advertising delivered from the cloud [6].

3. Market Share: It is very interesting that Android released its first phone in 2008, and only in four years, market shares for Android increased from 2% to 60%. Please see Table 1.1 [9].

4. Android has more options: Android is used as an operating system by many device manufacturers now like Samsung, Motorola, LG, HTC, Sony and so many other companies. Mobile Carriers like Verizon, AT&T, Sprint, T-Mobile, MetroPCS and Tracfone support Android. On the other hand Apple shares its operating system (OS) only with I Phone.

5. Android Play store: Google provides a play store to its end user, which has multiple applications, games, music, magazines, movies and TV programs, and books. It has a lot more options then Apple-store (I Phone) or Windows Market (Windows OS). Android also supports multi tasking of different activities.

6. User Friendly OS: One of the important reasons why Android is famous is because of its user friendly environment. Users can access each application easily and smoothly.
Table 1.1. World-Wide Smartphone Sales

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Windows Mobile</th>
<th>RIM</th>
<th>Symbian</th>
<th>IOS</th>
<th>Android</th>
<th>BADA</th>
<th>Windows Phone</th>
<th>Others</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2, 2012</td>
<td>N/A</td>
<td>5.2% / 1.9%</td>
<td>5.9% / 2.2%</td>
<td>18.8% / 6.9%</td>
<td>64.1% / 23.5%</td>
<td>2.7% / 1.0%</td>
<td>2.7% / 1.0%</td>
<td>0.6% / 0.2%</td>
<td>100.0% / 36.7%</td>
</tr>
<tr>
<td>Q1, 2012</td>
<td>N/A</td>
<td>6.9% / 2.4%</td>
<td>8.6% / 3.0%</td>
<td>22.9% / 7.9%</td>
<td>56.1% / 19.3%</td>
<td>2.7% / 0.9%</td>
<td>1.9% / 0.6%</td>
<td>0.9% / 0.3%</td>
<td>100.0% / 34.5%</td>
</tr>
<tr>
<td>2011</td>
<td>0.2% / 0.1%</td>
<td>10.9% / 2.9%</td>
<td>18.7% / 5.0%</td>
<td>18.9% / 5.0%</td>
<td>46.5% / 12.4%</td>
<td>2.0% / 0.5%</td>
<td>1.7% / 0.4%</td>
<td>1.0% / 0.3%</td>
<td>100.0% / 26.6%</td>
</tr>
<tr>
<td>2010</td>
<td>4.1% / 0.8%</td>
<td>16.6% / 3.1%</td>
<td>37.3% / 7.0%</td>
<td>15.6% / 2.9%</td>
<td>22.5% / 4.2%</td>
<td>1.2% / 0.2%</td>
<td>0.0% / 0.0%</td>
<td>2.6% / 0.5%</td>
<td>100.0% / 18.7%</td>
</tr>
<tr>
<td>2009</td>
<td>8.7% / 1.2%</td>
<td>19.9% / 2.8%</td>
<td>46.9% / 6.7%</td>
<td>14.4% / 2.1%</td>
<td>3.9% / 0.6%</td>
<td>N/A</td>
<td>N/A</td>
<td>6.1% / 0.9%</td>
<td>100.0% / 14.2%</td>
</tr>
<tr>
<td>2008</td>
<td>11.8% / 1.3%</td>
<td>16.6% / 1.9%</td>
<td>52.4% / 6.0%</td>
<td>8.2% / 0.9%</td>
<td>0.0% / 0.0%</td>
<td>N/A</td>
<td>N/A</td>
<td>11.0% / 1.3%</td>
<td>100.0% / 11.4%</td>
</tr>
<tr>
<td>2007</td>
<td>12.0% / 1.3%</td>
<td>9.6% / 1.0%</td>
<td>63.5% / 6.7%</td>
<td>2.7% / 0.3%</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>12.2% / 1.3%</td>
<td>100.0% / 10.6%</td>
</tr>
</tbody>
</table>


1.4 Research Objective

The main purpose of undertaking this research is to study the breathtaking progress that Android Operating System has made in first four years of its launch.

The research does not limit itself to what android is, but it also explains the concept of smart phones and their working. Android was launched in 2008 by Google Inc. as an open source mobile operating system. In 2012, it covers 60% of smartphone market globally, which is a remarkable progress. This has led Android Operating system to be the most used Operating System in Telecom industry. Recently, Android Play Store (formerly known as Google Market) celebrated 25 billion applications download.

As mentioned in the previous section, Android is open source. This allows any developer worldwide to tweak a single line of code; this line of code change can be either at application layer or at kernel layer. This research covers the launch of Android operating system followed by the emergence of smart phones in Chapter 1.
The next chapter describes Android in general, including its different version from Cupcake to Jelly Bean, and features and functions of each version. It also includes basic specifications of Android, architecture of Android and its layers.

The following chapter explains Android Booting-up process and Call functionality. This chapter also covers the details of Radio Interface Layer and Attention Commands (AT Commands). It explains in detail, how an android phone boots up, how we place a call and how we get an incoming call. By answering such questions I have explained how an Android smart phone works.

My next chapter, in this research focuses on logging and debugging mechanism of Android. We also discuss various logging tools and methods to capture the logs followed by the problems seen while using Android smart phones like call drops, Force Closes, ANRs, etc. We explain all these issues in depth with the logging mechanism of android. Live examples of root cause analysis of these issues are also explained in brief.

In the next phase we explain the upgrade methods and their significance in Android based smart phones. This chapter also discusses how the end user can get the latest flavor of android, without purchasing a new device, with a better User Interface, new features, and with bug fixes. FOTA (Firmware Over The Air) upgrade method is described in detail in this section. We also explain the real world examples of FOTA upgrade.

There are so many books and paper publications about Android and related applications but only few are related to Android telephony. The topics which are covered here like Android internals, Logging and Debugging mechanism and upgrade techniques for Android powered devices are important with respect to Android since enough information is not available about them. Our effort here provides certain framework to these practically relevant topics and their use while debugging, analyzing and fixing anything related to Android. This research will be extremely helpful to foster understanding among new android developers about Android Internals as well as everybody else who desires a general understanding of the internal working of an Android powered Smartphone.

Based on the reasons discussed above, I have chosen this research topic to understand and explain the internal working of Smartphone and Android Operating System.
CHAPTER 2

INTRODUCTION TO ANDROID

2.1 INTRODUCING ANDROID

Here, I will discuss brief history of Android. Android is an open-source Operating System created for mobile phones and other devices led by Google. In 2003, Andy Rubin and his team invented Android [8]. Later in 2005, Google acquired the Android operating system. Android is a software environment built for mobile devices. It is not a hardware platform. Android includes a Linux kernel-based OS, a rich UI, end-user applications, code libraries, application frameworks, multimedia support, and much more [8]. On the 5th of November in 2007, distribution of Google’s version was announced with the founding of the Open Handset Alliance. Open Handset Alliance is a group of device manufactures, Chipset Manufacturers and Mobile Carriers. Device Manufacturers include companies like HTC, LG, Samsung, Motorola and Sony. Chipset makers are Qualcomm, Texas Instrumental and Intel. Mobile Carrier companies are represented by companies like Verizon, Sprint, AT&T, T-Mobile, MetroPCS and lots of other companies all over the world [7]. Most of the Android platform is released under Apache 2.0 license. Google Android released the entire source code under an Apache license [8]. With Apache license, a user can freely download and use Android for personal and commercial purposes. It allows user to make changes to original software without having to contribute to the open source community.

2.2 FEATURES INTEGRATED IN ANDROID

Android has multiple features. Below are a few main features:

- Multiple language support: Android supports multiple languages from all over the world. It depends on device manufacturers but now most of the devices are provided with different languages like English, Spanish, French, Chinese, Korean, Hindi, etc. [10].

- Connectivity: Android supports most of the technologies for the mobile devices. Some of them are Global System for mobile communication (GSM), Code Division Multiple Access(CDMA), Long Term Evolution (LTE), Integrated Digital Enhanced Network (iDEN), Evolution Data Optimized (EVDO), Enhanced Data rates for
GSM Evolution (EDGE), NFC (Near Field Communications), Worldwide Interoperability for Microwave Access (WIMAX), and Enhanced Data rates for GSM Evolution (EDGE). Using above mentioned technologies, Android provides connectivity across the world [10].

- **Java support**: Android applications are made in Java. However, Android doesn’t support JVM (Java Virtual Machine). Android has its own virtual machine for the compilation; it is called Dalvik Virtual Machine (DVM). It is basically made only for Android Operating System. Java classes are compiled and executed in Dalvik Machine [10].

- **External storage**: Android supports external memory slots as per device manufacturer’s requirements. Most Android devices include microSD slot and can read microSD cards formatted with FAT32, Ext3 or Ext4 file system. To allow use of high-capacity storage media such as USB flash drives and USB HDDs, many Android tablets also include USB 'A' receptacle. Storage formatted with FAT32 is handled by Linux Kernel VFAT driver, while 3rd party solutions are required to handle other popular file systems such as NTFS, HFS Plus and exFAT [10].

- **Bluetooth**: Android platform supports various kinds of Bluetooth protocols for transferring, sending, and receiving files, pictures, accessing the phone book, vcard, and schedules. It supports A2DP, AVRCP, sending files, accessing the phone book (PBAP), voice dialing, and sending contacts between phones [10].

- **Messaging**: Android also supports Short Messaging Services and Multimedia Message service (MMS). MMS service includes sending image, video, audio, calendar events, contacts, locations in form of Messages. Android supports threaded messages [10].

- **Media support**: Media support depends on device manufacturers and the versions of Android. Most of the Android versions support various audio/video/still media formats like Motion Picture Experts Group Layer3 (MP3), Waveform Audio File Format (.WAV), Musical Instrument Digital Interface (MIDI), Joint Photographic Experts Group (JPEG), Portable Network Graphics (.PNG), Graphics Interchange Format (.GIF), bitmap image file (.BIT), 3GPP file format (.3gp), Moving Picture Experts Group (MPEG), Adaptive Multi-Rate Codec File, and Advanced Audio Coding (AAC) [10].

- **Streaming media support**: Streaming media support also depends on device manufacturers and different Android versions. Most of the Android versions support Real time streaming (RTSP) and HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic streaming are supported by the Flash plug-in [10].

- **Voice based features**: Voice based features like calling, browsing, and accessing Navigation are supported by Android. The latest versions of Android like Ice cream Sandwich (ICS), and Jelly-Bean, both give better voice command and smooth functionality [10].
• Multitasking: Executing more than one application at the same time is supported by most of the Android devices. Example would be while downloading a file while listening to music. Multitasking of applications and specific memory allocations as per activity handling are available in Android. This leads to better utilization of resources [10].

• Tethering: Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi hotspot. It also supports sharing the files via applications like smart WIFI or smart share [10].

• Screen capture: Android supports screen capturing by different types of combinations as per requirements from device manufactures, like volume up + Power button or volume down + power button. We can use Dalvik Debug Monitor Server (DDMS) to capture the screenshot. We just need to connect the device to Dalvik Debug Monitor service (DDMS) via USB cable [10].

• Video calling: In the earlier versions of Android, it did not support video calling. After the Gingerbread version, device manufacturers also provided front end camera and back camera on the same device. Video calling through Google Talk is available in Android 2.3.4 and later. Some applications like Skype and qik video chat, etc also provide the video calling functions in smart phones [10].

• Web browser: Android supports the browser based on WebKit layout engine. In earlier versions of Android, devices supported 4 to 8 tabs in a browser. But in Ice-cream Sandwich or Jelly-Bean, a device can open 16 tabs at a time. Google Chrome is also supported now as a browser [10].

• Storage: To give better performance, Android uses SQLite database for storage as storage type. SQLite is the light database [10].

2.3 Android Platform Versions

Android is a milestone in the smart phone world. The first phone was released in 2008. Cupcake is the first official version of Android. After that Google provide a few versions of Android in a certain time period. Each version improves bug fixes, adds new features and functionality, and improves the performance. Each version is named after a dessert and is going alphabetically. The following is a review of all Android version and their basic features:

• Android Version 1.0: It was the first version released by Android. There was no platform highlights for this version [11].

• Android Version 1.1: It was released in February 2009. GI from T-Mobile was the first device from this version. It supports new features like Maps, Dialer, added plug in to save attachments in Multimedia Message Service (MMS), and provides support in layouts [11].
• Android Version 1.5 (Cupcake): It was released in April 2009. It has many new features for end users as well as for developers. Its improvements are the camera performance, Global positioning system (GPS), browser access and Gmail conversation. This version adds new features like on screen soft keyboard, home screen widget, video recording and playback, hands free experience and auto pairing for Bluetooth, a few functions in a browser like copy & paste, call log history, SD card file system, and upload photo/video to YouTube/Picasa [11].

• Android Version 1.6 (Donut): It was released in September, 2009. It has many new features such as quick search box, improved camera and camcorder, a facility to add Virtual Private Network (VPN), new accessibility services, updates for Google play, text to speech engine, gestures, and technology support for code division multiple access (CDMA) [11].

• Android Version 2.0/2.1 (Éclair): It was released in October, 2009. This version adds multiple features like supporting synchronization for more than one account, Exchange email support, search functionality for messaging, auto delete for old messages, built in camcorder flash support, scene mode digital zoom, Android virtual keyboard, Hyper Text Markup Language (HTML) 5 for browser, and it supports various Bluetooth profiles [11].

• Android Version 2.2 (Froyo): This version was released in May, 2010. It supports various features like tip widget on home screen, added short cuts to home screen, improved security, auto discovery for exchange email, easy access and LED flash to camera/camcorder, portable WIFI hotspot function, improved overall performance, multiple language support for Android keyboard, kernel upgrade for more memory [11].

• Android Version 2.3 (Ginger Bread): This version was released in December, 2010. It is the major version for an Android. It provides lots of new features to the end user like better user interface, faster text input, one touch word selection, improved power management, control over application, Internet calling, Near Field communication (NFC), Native input and sensor event, Gyroscope, improved 3dimension motion, better graphics management, support new media format, and access to multiple cameras (back end/front end) [11].

• Android Version 3.0 (Honeycomb): This Version was released on February 22, 2011 [12]. It was made only for tablets. Motorola Xoom is the first tablet on Honeycomb. It provides New User Interface (UI), system bar for global status and notification, customizable home screens, improvements in text selections, multiple tabs in browser, widgets, high performance, digital media transfer, HTTP live streaming [11].

• Android Version 4.0 (Ice-cream Sandwich): This version was released on October 19, 2011. This version is used for both phone and tablets. It gives easy multitasking, rich notification, customizable home screen, resizable widgets, home screen folders, new lock screen functions, swipe to dismiss notification, tasks and browser, improved text input and spell checking, powerful voice input engine, easy accessibility, control over network, visual voicemail, better camera performance, sharing the screenshots,
live effect for transforming video, improvements in email application, Face unlock system, Android Beam for near field communication and WIFI direct [11].

- Android Version 4.1 (Jelly-Bean): This version released on July 19 2012 [13]. It is the smoothest version of Android. It gives the multiple features which includes faster response, gesture mode for easy accessibility, Android beam, better animations in browser, various functions in camera/camcorder, more language support, accurate keyboard, new notification style, improved search performance in people application, new voice command, more options in Google now and in Google search [13].

### 2.4 CURRENT DISTRIBUTION OF ANDROID VERSION

According to the descriptions we have seen above, Version 1.1 is the first version of Android, but Version 1.5 was the first version that legally released as the first version and launched first handset with Cupcake version. Please Table 2.1 [14] for current distribution of Android Versions.

<table>
<thead>
<tr>
<th>VERSION</th>
<th>Code Name</th>
<th>API</th>
<th>DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>Cupcake</td>
<td>3</td>
<td>0.10%</td>
</tr>
<tr>
<td>1.6</td>
<td>Donut</td>
<td>4</td>
<td>0.40%</td>
</tr>
<tr>
<td>2.1</td>
<td>Éclair</td>
<td>7</td>
<td>3.40%</td>
</tr>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>12.90%</td>
</tr>
<tr>
<td>2.3.X – 2.3.7</td>
<td>Gingerbread</td>
<td>9,10</td>
<td>55.80%</td>
</tr>
<tr>
<td>3.1 , 3.2</td>
<td>Honeycomb</td>
<td>12,13</td>
<td>1.90%</td>
</tr>
<tr>
<td>4.0.3 – 4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>23.70%</td>
</tr>
<tr>
<td>4.1</td>
<td>Jelly Bean</td>
<td>16</td>
<td>1.80%</td>
</tr>
</tbody>
</table>


### 2.5 ANDROID ARCHITECTURE

Android was released under two different open source licenses. The Linux kernel is released under the General Public License (GPL), and Android platform excluding Linux kernel, is licensed under the Apache Software License (ASL). They both are balancing Android as an open source. Android OS has a multi-layered architecture, which is shown in Figure 2.1 [15]. These layers can be broadly described into the following layers:

- The Application layer: All the applications which the user interacts with directly are part of the application layer. This includes both core and third party applications like phone, contacts, games, home, browser, etc. [8].

- The Application Framework layer: This layer provides the framework to support all the processes, activities and events in the phone. It provides Activities and views,

telephony services, Resources; Location-based services. It also helps in managing all the resources for the OS on the device [8].

Libraries and Android Runtime: For all the events/activities/processes to work, libraries provide the base. It includes browser technology from Web Kit, database support via SQLite, Advanced graphics support including 2D, 3D, animation from
SGL, audio and video media support from Packet Video’s open core, and SSL capabilities from the Apache project [8].

- The Android runtime provides the core java libraries for full features Java programming and the DVM (Dalvik Virtual Machine), which employs service of the Linux-based kernel to provide an environment to host Android application [8].

- The Linux kernel Layer: This layer provides core services like process, memory and file system management. It is also responsible for hardware specific drivers such as Wi-Fi and Bluetooth. This layer is designed to be flexible with many optional components that largely rely on the availability of specific hardware on a given device. It includes a few features like touch screens, cameras, GPS receivers and accelerometers [8].
CHAPTER 3

ANDROID INTERNALS

3.1 BOOT UP PROCESS IN ANDROID

The boot up process can be described as the process of powering on the device [16]. During boot up process device goes through these various stages [17].

3.1.1 Stage 1: The Android Boot Process from Power On

When Android powered device is boot up, Boot ROM code initialize the hardware and locate the boot media with the help of Central Processing Unit (CPU). When device hardware is initialized, ROM code find the boot media. It also copies the boot loader into internal RAM. After this step, the process of boot loader is started. Figure 3.1 explains the first stage, when device is booting up [16].

Figure 3.1. Initial power up.
3.1.2 Stage 2: The Boot Loader

Boot loaders (see Figure 3.2) are combinations of instructions to boot any operating system kernel, generally speaking a boot loader is a small program that loads the operating system into the computer’s memory when the system is booted, and also starts the operating system. In Android the BL (Boot loader) is a program which is special in its own ways. This is because Boot Loader is separate from Linux kernel, which is used to setup initial memories and load kernel to RAM [17].

![Figure 3.2. Boot loader.](image)

Figure 3.2 explains the third stage of Boot loader. Details of each stage are explained below:

- During step-1, Initialization of boot loader takes place. Boot loader sets up External RAM for further processing [16].
- In step 2, Boot loader makes sure when external RAM is available and system is also ready to run specific process [16].
- In step 3, the first major program will run. It includes important activities from setting up additional memory, to setting up file system. In an Android powered smartphone,
it also supports loading code for the modem CPU and low level security aspects and memory protections as well [16].

• In step 4, Boot loader will look for a Linux Kernel when a specific task is over for boot loader. As per requirements of device manufacturer, it will load this from the boot media and place it in the RAM. It will place certain boot parameters in memory for the kernel when it initializes [16].

• Finally, Linux Kernel will start its operations [16].

3.1.3 Stage 3: The Linux Kernel

Linux Kernel is a heart of the Android operating system, the whole process remains the same, as the Linux kernel starts up, and it sets up all the necessary initializations which are needed by the system. For instances, it initializes the interrupt controllers, sets up memory protections, caches and schedule processes. In stage -4, the Linux kernel initializes all the memory units. Once this is done, the system is ready to execute the user space processes. User space processes refer to the processes that are executed in the user space e.g. Applications, user data. To launch the user space processes, kernel must execute init process first. This is found in system/core/init. The init process initializes all the user space processes [17].

3.1.4 Stage 4: The Init Process

The init process is the "head of all the processes". This is because all other processes in the system are launched from this process or they are one of its descendants [17].

Init.rc: The init.rc is a script that is looked up by the init process. The init.rc script is placed in system/core/rootdir in the Android open source project. This script describes all kinds of system services, file systems, and the parameters that need to be set up. The init process parses this script and launches the system service processes [17].

3.1.5 Stage 5: Zygote and Dalvik

Once the init process launches the system services, then it starts launching the zygote process. This means it will just start, initialize and execute the Dalvik Virtual Machine (DVM) [17].
3.1.6 Stage 6: The System Server

The first java component is the system server that will run in system. It is responsible for starting Android services such as telephony manager and Bluetooth. In the Android operating system start up of each service is currently written directly into the run method of the system server. The system server source can be found in the file frameworks/base/services/java/com/android/server/SystemServer.java in the open source project.

Figure 3.3 explains the Interaction of system server with the File System [16].

![System server diagram]

Figure 3.3. System server.

3.2 INTRODUCTION TO THE ATTENTION COMMANDS (AT COMMANDS)

AT commands are strings of data which begin with the prefix AT and can be sent to a communication module for the purpose of programming [18]. In other words, to communicate with Radio Interface Layer (RIL) which is the modem or in simpler words the base band side of the phone we need the AT commands. AT commands are related to asynchronous serial interface and we can use Hyper Terminal tool/software in Windows to send it. If module is connected to Personal computer/machine using RS232 (Electric Standard), then also using RS232, can also send the message[18]. AT commands were introduced in 1977 by Hayes Communications Company for dial-up modems [18]. AT
Commands should communicate with many different areas. Overall, AT commands are used to control Modem functionality. The supported categories include:

Some Important AT commands:

- **AT+CGMI**: Request Manufacturer Identification: execution command returns the device manufacturer identification code [19].
- **AT+CGMM**: Request Model Identification: execution command returns the device model identification code [19].
- **AT+CGMR**: Request Revision identification: Execution command returns device software revision number [19].
- **AT+CGSN**: Request Product Serial Number Identification: Execution command returns the product serial number, identified as the IMEI (Identity Mobile Equipment Identity) of the mobile [19].
- **AT+CIMI**: Request International Mobile Subscriber Identity: Execution command returns the value of the Internal Mobile subscriber Identity stores in the SIM (Subscriber Identity Module) [19].
- **AT+CHUP**: Hang Up Call: Execution commands cancel all active and held calls, also if a conference call is going on [19].
- **AT+CRS**: Cellular Result Code: set command controls whether or not the extended format of incoming call indication is used [19].
- **AT+CNUM**: Subscriber Number: Execution command returns the MSISDN (Mobile Subscriber Identity Number) only when the phone number of the device has been stored in the SIM card [19].
- **AT+CCFC**: Call Forwarding Number and Condition: Execution command controls the call forwarding supplementary service [19].
- **AT+CCWA**: Call Waiting: Set Command allows the control of the call waiting supplementary service like activation, deactivation and status query [19].
- **AT+CLCC**: List Current Calls: Execution command returns the list of current calls and their characteristics [19].
- **AT+CPAS**: Phone Activity Status: Execution command reports the device status [19].
- **AT+CPIN**: Enter PIN: Set command sends to the device a password which is necessary before it can be operated like SIM pin [19].
- **AT+CSQ**: Signal Quality: Execution command reports received signal quality indicators [19].
- **AT+CPBS**: Select Phonebook Memory Storage: Set command selects phonebook memory storage, which will be used by other phonebook commands [19].
• **AT+CBC**: Battery Charge: Execution command returns the current battery charge status [19].

• **AT+CSMS**: Select Message Service: Set command selects messaging service. It returns the type of message [19].

• **AT+CMGL**: List Messages: Execution commands reports the list of all the messages with status value stored into message storage [19].

### 3.3 Radio Interface Layer in Android

Above we have briefly described the AT commands, and now let’s have a look at the RIL. There is a Radio Layer Interface in Android and it is about telephony service.

Figure 3.4 [20] gives us a snapshot of the calling flow mechanism of Android. It displays vital components of RIL, which is the heart of any smart phone for substantial calling and Data activities.

Below I have tried to explain the general working of telephony framework when communicating with modem in the Android platform:

1. The modem receives all the inbound requests and processed data delivered by the framework through the RIL [21].

2. The RIL represents the Radio interface integrating modem and Android telephony [21].

3. Certain inbound events concerning network originating at the modem are also interfaced by the RIL with the Telephony framework [21].

The RIL has two parts known as RIL daemon and vendor RIL.

RIL daemon is the interface between Android telephony framework and vendor RIL. All requests originating to and from the RIL are stored in one interface file RIL.java which is a part of the Android Telephony network. Customized implementations for the vendor RIL as implemented by each vendor are stored in a shared library. Modem specific vendor RIL are usually used by the vendor.

Basic Boot up sequence with reference to telephony is as under:

1. Android telephony framework is started as soon as the phone is powered on and the phone processes begin.

2. A socket connection is made between the RIL daemon and framework followed by the ril daemon search for the vendor ril path from the system property.

3. Vendor RIL is then loaded as the shared library.

4. The RIL_INIT method is employed by the vendor RIL by calling it during its initial start.
5. Telephony framework receives references for all vendor RIL functionalities from the RIL daemon, which does so, by calling the RIL_Register

RIL Interaction: There are two forms of communication that the RIL handles:

- Solicited commands: Solicited commands comes from the upper application framework layer like dial up, sending SMS and read/write contact/SMS from SIM. The RIL daemon fetches the all commands from application framework layer to vendor RIL. Then, with the help of inter process communication/AT commands, vendor RIL interacts with modem protocol stack. In respond, modem sends the commands to RIL vendor through IPC/AT commands. Then RIL vendor send the response to application framework [21].

- Unsolicited Commands: The RIL can receive these types of commands/events like receiving SMS, and network state changes at any time from the radio hardware.
These are dispatched by the RIL and sent back to the Android framework. For instance, receive an incoming call (from RIL-daemon to telephony service) [21].

The interesting part here to check is how does the incoming call ring? To understand this we need to understand two term, MO and MT. MO stands for mobile originated call, which means the call is originated by the mobile station, or in other words you make a call. The MT call stands for mobile terminated call, which means the call is an incoming call from a different mobile station or a base station.

For an unsolicited call, once the RILd determines the state of the call, it uses the comprised set of API’s which is the AT command (AT+CLCC) to check for any incoming calls. The function of this AT command is to return the list of MT calls. All communication with radio hardware is processed by the radio specific RIL vendor. Unsolicited commands are then used to call RIL Daemon (RILd) by the radio specific RIL vendor. The vendor RILd receives unsolicited commands by the Radio (protocol stack) through Inter Process Communication (IPC).
CHAPTER 4

THE LOGGING AND DEBUGGING MECHANISM

4.1 INTRODUCTION

In this chapter we will review the logging and debugging mechanism. In the android environment on a phone, many functions may be broken like any other software platform.

Issues need to be addressed and fixed for a technically sound software platform and a better user experience. Therefore, from the troubleshooting development perspective, logging is a very important tool. The Following paragraphs describe about the anatomy and severity of logs. They also provide details of what information is contained within these logs, methods to pull relevant logs from the device, filtering required information from these logs and other such details.

Logging is the method to detect different kinds of issues. In its simplest form, logs allows developer to deduce the root cause of an issue based on the activity running in the background and see the exact time the issue take place. After getting all the required information, the developer fixes the issue for the next software version. This subsequently, produces a good user experience subsequently and a low priced product for the end user. We will see the tools, methods and common issues involved with logging in this chapter.

Android based Smartphone contain Application side and the Baseband side. Application side supports components such as camera, messaging, call, etc., and Baseband related components include antenna, modem, etc. Android can only provide for Application side logging capabilities as the Baseband logging is usually proprietary to the respective manufacturing companies like Qualcomm INC., Intel, etc. To reduce the complexity of data and the logging nuances associated with Baseband logging we will keep our logging methods focused on basic Android logs and related basic logging tools.

4.2 TOOLS USED TO CAPTURE LOGS

This section describes the Android tools and their variants to capture logs. Logs are real-time monitoring records running in the background which are stored in files on the
phone while the user performs activities. When an unusual behavior occurs, it is recorded in real-time on the phone. After debugging and analysis of the logs, the developer can see the root cause of the issue, the time when the problem occurred, which application caused the issue, as well as what other processes were running at the time when the issue occurred. There are several tools and debuggers used in business but the basic tools in Android are the Android Debug Bridge (ADB) and Dalvik Debug Monitor Service (DDMS).

4.2.1 Dalvik Debug Monitor Service (DDMS)
This is a tool which comes along with the Android Software Development Kit (SDK). Dalvik Debug Monitor Server (DDMS) is also integrated in eclipse. It provides screen capture service, thread information of device, logcat process, radio information, call and short message service spoofing and so many other functions. DDMS works with both emulator and a connected device. If both are connected and running simultaneously, DDMS defaults to emulator. In Android, each application runs its own process on its own virtual machine (VM). Android Debug Bridge (ADB) is connected to Dalvik Debug Monitor Server (DDMS), when it starts which basically means when a device is communicating to a machine, one type of service called as Virtual Monitor is created between ADB (Android Debug Bridge) and Dalvik Debug Monitor Service (DDMS). It is responsible to notify when application on the device is started or terminated [22].

4.2.2. Android Debug Bridge (ADB)
It is a powerful command based utility which allows communicating with an emulator instance or connected Android device. It has three commands. First is a client which runs on development machine, and second is a server which runs in a background. This manages the communication between client and ADB daemon and third is a daemon which runs as background process in each device instance. Android Debug Bridge (ADB) is also responsible for managing the current status of a device, running shell commands on it, managing port forwarding, and copy files to and from device [23].

Figure 4.1 shows us the can work flow of ADB. First, ADB client issues the command from the machine to start the ADB server. Then the ADB server set up the
communication or connection with Android powered device by scanning the port. Finally, after the connection is established, ADB commands can run on the system.

### 4.2.3 General ADB Commands

- Installing an Application:
  - `adb install <path_to_apk>` (this command install files to `/system/app` or `/data/app`) [23].

- Copying Files to or from a Device Instance:
  - Unlike the install command, which only copies .apk file to a specific location, the pull and push commands let you copy arbitrary directories and files to any location in an emulator/device instance [23].
  - To copy a file or directory (recursively) from the device, use:
    ```
    adb pull <remote> <local>
    ```
  - To copy a file or directory (recursively) to the device, use:
    ```
    adb push <local> <remote>
    ```

- Stopping the `adb` Server:
• if adb does not respond to a command, you can terminate the server and restart it and that may resolve the problem [23].

• To stop the adb server, use the `adb kill-server`. You can then restart the server by issuing any adb command [23].

• Logcat:
  
  • The Android logging system provides a mechanism for collecting and viewing system debug output. Logs from various applications and portions of the system are collected in a series of circular buffers, which then can be viewed and filtered by the logcat command. You can use logcat from an ADB shell to view the log messages [23].

### 4.3 METHODS TO CAPTURE THE LOGS

In this section we explain the methods to capture the logs from a device. There are two ways to capture the logs:

• Dalvik Debug Monitor Server (DDMS)

• ADB commands.

Android provides lots of facilities for debugging purposes. To capture the logs, we have to connect the mobile device to a laptop using USB Cable. Other thing we need to make sure is that the device has “USB Debugging” option enabled.

#### 4.3.1 Capturing Logs Using DDMS

Two of the most used options in DDMS are Screen Capture and Logcat logging.

#### 4.3.1.1 SCREEN CAPTURE

We can capture the screen on the Mobile Device using screen capture option in Dalvik Debug Monitor Server (DDMS). This software is located in the Android SDK. As shown in Figure 5.1 (p. 33), it is under Device->Screen Capture. This will capture whatever is currently displayed on the Mobile Device. This can be saved as .png or .jpg file for debugging. After looking at the saved image, a developer will know what problem was encountered, the exact error pop-up message, and UI distortions [22].

Figure 4.2 shows the screen capture option using the Dalvik Debug Monitor Server (DDMS).
4.3.1.2 LOGCAT

After connecting the android device we need to select device-> logcat option to see what processes are running at the time and, exactly what activities are going on at a certain time. When a Logcat process runs in Dalvik Virtual Monitor Server (DDMS), logs are running in the background of the device for different applications and are collected in a particular buffer.

Figure 4.3 shows sample logs captured by Logcat process. This is helpful to developers to understand the exact nature and root cause of the problem.

4.3.2 Capturing Logs Using Android Debug Bridge (ADB)

There are many types of logs but main log, radio log, kernel log, and event log are the basic logs which are very important for debugging and analysis of abnormal behaviour of the device [24]. We will see how to filter log output and use commands to capture the logs.

4.3.2.1 FILTERING LOG OUTPUT

Each Android log message has a tag and priority associated with it. The tag of a log message is a short string indicating the system component from which the message
originates. The priority is indicated by one of the following character values, in order from the lowest to the highest priority:

- **V**: Verbose (lowest priority): Zero level information that end user can identify
- **D**: Debug
- **I**: Info
- **W**: Warning
- **E**: Error
- **F**: Fatal
- **S**: Silent (highest priority, on which nothing is ever printed) [24].

### 4.3.2.2 Commands to Capture the Logs

1. **Main Log**: The first basic log in the buffer memory is used to reflect acitivities that are running in the phone. Main log has information related to applications. It contains basic information only. Consider one example: a user switches to the camera and takes one photo, then press home key. Main logs contain all the information related to camera [24].

   Command: `adb logcat -b main>main.txt`

2. **Radio Log**: This log contains the information related to telephony/radio messages. It will provide some more information for baseband/modem side. Radion log contains
any network side issue or call related problems. Consider one example: A user is browsing websites on the device, and suddenly the device fails to open a web page. At that time Radio log contains the information about network loss [24].

Command: adb logcat –b radio>radio.txt

3. Kernel Logs: It provides the information related to kernel side activities. It contains important information about the communication between kernel layers, modem and application framework in Android device. Consider one example: navigation running on the device and it crashes unexpectedly. At that time kernel log is helpful to analyze the issue [24].

Command: adb logcat kernel>kernel.txt

4. Event Log: It provides information about launched or finished activities. It contains the information related to any event which occurs in the phone. Consider an example: a user change the mode of the device from landscape to portrait, but mode does not change. Event log contains all the information related to it [24].

Command: adb logcat –b event>event.txt

### 4.4 Commonly Seen Android Errors

In this section, I will list some issues that are encountered during our daily use of Smartphone. Our objective here is to answer the most commonly observed issues related to network, applications and other core internal systems like the kernel.

#### 4.4.1 Call Drops

We always have call drop while in transit in urban or sub urban areas or even in stationary conditions. Now my objective here is to educate the user how these errors actually look when we extract a log. We will discuss more about log variants in our next section, but let’s focus now on the above issue.

Device using regular network:

```
09-19 04:39:17.327 D/RILJ (  480): [UNSL]<
UNSOL_RESPONSE_CALL_STATE_CHANGED
09-19 04:39:17.334 D/RILC (  153): rild: grabPartialWakeLock, s_wakeup_lock_flag - 1
09-19 04:39:17.358 D/RILJ (  480): getDlcRequestId - Request : LAST_CALL_FAIL_CAUSE, Dlc id – 2
```
The above log gives me a description of the dropped call due to the network error. The point that arises is how do we know about this network issue. In the last chapter we studied about AT commands, RIL, Rild (ril daemon), unsolicited, and solicited commands. If we look at log snippet then we can observe that the call got disconnected from the network was in unsolicited state, and immediately the rild came in action and caused a wake to the device; by wake here we meant the backlight of the device turns on:

09-19 04:39:17.358 D/RILJ (  480): DLC [2]: Tx [6677] sent = 77 answered = 76 Tout = 0
- - - LAST_CALL_FAIL_CAUSE: 13000ms
09-19 04:39:17.358 D/RIL (  153): lge-ril:
runRequestFunction(LAST_CALL_FAIL_CAUSE) s_InitMutex LOCKED. dlc index - 2,
DLC - 2, DLC create tracker - 0xbf
requestInterface(LAST_CALL_FAIL_CAUSE) invoking *requestFunction(req:18,
data:00000000, len:0)...
failure" : fail-cause = temporary failure

The log shows that communication between Vendor RIL and Ril daemon got interrupted and created the call failure error. Above code in red color is generating the error report for temporary failure of network.

### 4.4.2 Force Close on Gmail Application

This issue relates to Gmail force close. It shows that FATAL exception has occurred in main thread. Also, Runtime Exception fails to destroy the background activity and enough memory is not available. It causes the Gmail Force Close. The issue clearly tells us by looking at the logs that the Gmail widget has gone through a fatal exception in the java.lang package which in turn caused the exception to arrive for java.lang.NullPointerException.

07-23 21:45:48.726 E 22165 AndroidRuntime FATAL EXCEPTION: main
07-23 21:45:48.726 E 22165 AndroidRuntime java.lang.RuntimeException: Unable to destroy activity
{com.google.android.gm/com.google.android.gm.GmailActivity}:
java.lang.NullPointerException
07-23 21:45:48.726 E 22165 AndroidRuntime Caused by:
java.lang.NullPointerException
07-23 21:45:48.726 E 22165 AndroidRuntime at
android.widget.AbsListView.setItemChecked(AbsListView.java:1411)
4.4.3 Black Screen on Device

Sometimes while using the Smartphone, we get the black screen on the device. There are certain reasons for black screen like multiple activities are running at a time, or heavy application is running in the background of the device. Below logs contain the information about black screen issue while using the phone and playing Pandora radio at same time.

05-12 12:49:57.579 I/ActivityManager(  245): process name to start: com.android.vending -> shows that the device is in low memory state that the LMK(Low Memory Killer)kills the processes with low adj down to 4, which tells that the low memory state is quite severe
<4>[29770.487224] select 24727 (equicksearchbox), adj 7, size 4122, to kill
<4>[29770.487254] send sigkill to 24727 (equicksearchbox), adj 7, size 4122
<4>[29771.923986] select 24708 (vvm.application), adj 7, size 3967, to kill
<4>[29771.924016] send sigkill to 24708 (vvm.application), adj 7, size 3967
<4>[29778.548599] select 957 (obile.bonusapps), adj 4, size 3934, to kill
<4>[29778.548629] select 966 (angable.weather), adj 4, size 7840, to kill

The log shows that the device was in low memory state, and it caused Low memory killer to kill the running processes frequently, which caused the black screen symptom in this issue. Since the device has limited resource, and the Pandora radio uses quite a lot of memory when it's running, the device may get to low memory state more easily when it has the Pandora radio app on its background. All one can do is extra apps to ensure this issue doesn’t persist.

4.4.4 Application not Responding on Facebook (ANR)

Application not responding is the common issue seen on every smartphone. When any application is not giving proper response within about 5-7 seconds, ANR pop up is generated on that application. Please see below an example of Application not responding error on Facebook:

08-25 12:16:50.797 E/ActivityManager(  436): ANR in com.facebook.katana
08-25 12:16:50.797 W/ActivityManager(  436): Killing ProcessRecord {4530919028175:com.facebook.katana/10022}: background ANR
08-25 12:16:50.847 I/ActivityManager(  436): Process com.facebook.katana (pid 28175) has died.
We can see ANR is generated in com.facebook.katana in first line. Android memory management gives the priority to all the processes. As shown in above snippet, Activity manager kills the process to prioritize, and ANR occurs on the device.
CHAPTER 5

UPGRADE METHODS FOR ANDROID DEVICES

Android operating system for mobile devices has become a common place name in recent time. The progressive software innovations associated with it have led to the development of solutions for both cell phones and tablet devices. It all started with Apple pie, the first version of Android, to the latest version named Jellybean. Android has a whimsical quality of naming its software upgrades alphabetically after dessert items like Cupcake, Donut, Éclair, etc. This topic is discussed in this section, explaining what an upgrade for the mobile world is, why it is necessary, different types of upgrades and how they work. Since Software upgrades form an integral part of Android software ecosystem, the sections that follow describe upgrades, their types, and fundamental operation.

5.1 WHAT IS AN UPGRADE OR UPDATE FOR ANDROID DEVICES

An upgrade provides the user with bug fixes of all minor and major categories. This might improve performance and functionality of the devices. Most software fixes are sent over the air to the devices. The process includes alerting the user to the availability of upgrade software on the device. This update changes the software version and makes all the changes associated with the software update. It is the product of combined efforts from mobile carrier, device manufacture, device model, Country and unique id of device.

5.2 WHY IT IS NECESSARY

The software developer (Google Android side and device manufacturer side) adds new features, user interface enhancement, graphics improvements and other bug fixes with each new version. If the end user wishes to implement these improvements on the device, they can do this via the upgrade process.
5.3 Different Types of Upgrade

There are 3 main upgrade methods for upgrade/update:

- FOTA (Firmware Over The Air)
- GOTA (Google Over The Air)
- Web Upgrade via Personal Computer

5.4 How does FOTA work

The purpose of FOTA is to upgrade the firmware on devices with safety and security. It is better to send the updated or added code segments instead of the whole firmware package due to limitations based on bandwidth and other restrictions. Bricked phone is the ultimate consequence of a FOTA update gone awry; therefore, FOTA is a high risk balancing act between optimum output and security [25].

FOTA process contains two parts:

1. DIFF generator – This is used to locate the distinction within two consecutive or rather any separate versions of code like v3 and v4.
2. The DELTA – This has all code segments necessary to upgrade the device and the information on processing, but is usually much smaller than the actual firmware code for the target version (in the case mentioned above - v4)

As we can see in Figure 5.1 [25] software is upgrading from v3 (older) to v4 (newer) version. First, the DIFF computes the differences between the two versions. Then it applies multiple compression and optimization algorithms. After this, it adds the key for package security and integrity. The final step is applying the compressed and optimized patch from v3 to v4.

The firmware update cycle can be defined in four parts as following:

1. The generation of the firmware update package referred to as Delta by the handset manufacturer: The size of the firmware update needs to be optimized for bandwidth and other related restrictions. Delta generation technology is a FOTA software that is vendor proprietary and should be compatible or form a close association with the hardware and software aspects of the handset. It is usually licensed by the vendor and used by the manufacturer of devices. Delta package has all the necessary information about the handset like information on manufacturer, model, source, target firmware version, security descriptors and the instructions for the update [26].

2. Transmission and submission of the update to a device management server: Once the Delta is generated, it is submitted to a firmware management server. The firmware management is responsible for complete management of the firmware update procedure. This server coordinates and receives the update packages which are
coming from different sources. It also delivers the packages to the device over the air. The firmware management server is typically hosted by device manufacturer and mobile operator or by third party [26].

3. Downloading of the firmware to the device: Typical duties of the client software for device management include receiving the package, coordinating interactions between the user and the server, forwards the package to the agent that manages updates, and notifies the update status to the server. Firmware download can be differentiated into two parts that include client initiated and servers initiated [26].

4. Updating of the device firmware by the update agent: In the final stage of the upgrade process, the update agent applies updates and replaces the current software firmware image with a new software firmware. The unpacking of Delta and parts of device memory that need to be changed is controlled by the update agent. It needs to done successfully without any data or memory loss or any other issues. The update agent must be written by device manufacturer or FOTA vendor for device manufacturer. A core algorithm engine that is FOTA vendor proprietary is included in the update agent. When coupled with Delta generation technology, this is given out by the special FOTA vendors like Red Band and Innopath [26].

### 5.4.1 DELTA Definition

DELTA is the most important part of the FOTA process. FOTA helps to change the older software version to newer software version. It is an algorithm which contains the code with all the necessary information about source version, target version and code which is used to convert from source version to target version.
5.4.2 Challenges in Development DELTA Technology

1. Once the update has started, it has to be completed successfully without any failures [26].

2. All handset have limitations with their memory, processor speed, and the speed of memory access [26].

3. The biggest challenge to DELTA generation between software firmware versions is the ‘ripple effect’ of reference changes. It means that just a little change in the firmware image will result in a cascade of consequential reference changes and shifts in image blocks. As firmware code and data shifts from one image to another, all references change. This creates a large consequential amount of change throughout the image [26].

5.5 FOTA Upgrade through Real Scenario

Now we will consider the real scenario of FOTA upgrade.

Step 1: Here the FOTA package is pushed from the FOTA server to device end. We can see in Figure 5.2 the software package available on device side and ready to download. The device is communicating with server at that time.

Step 2: In Figure 5.3, software package is downloading in the background of device. Users can do anything on their device while package is downloading in the background.

Figure 5.2. Showing package on device.
Step 3: After downloading of software package is complete on the device under test, the next step will occur. The next step will be to update the software package. As shown in Figure 5.4 update option pop up on the device.

Step 4: After selecting the update option, the device will start update process automatically as shown in Figure 5.5. The whole process will take around 2-10 minutes
depending on the package size. If the package size is less, then it will take less time, whereas, if the package size is big, then it will take more time than usual.

Step 5: In software updating process, the device communicates with FOTA server until the process is completed. Now Delta technology takes part in the whole communication and changes the software version from older one to newer one as mentioned in firmware update cycle. After the process, the device gets notification of a successful update from the older software version to newer software version.

### 5.6 HOW FOTA UPGRADE WILL SAVE MONEY

FOTA is a commercially proven technology with a significant market momentum. Below are the key benefits of deploying the FOTA.

- FOTA eliminates software based device recalls [25].
- FOTA gives bug fixes on the device [25].
- FOTA improves time to market for new services[25].
- FOTA reduces customer service and warranty costs [25].
- FOTA avoids warranty actions like bringing in a device to solve a problem that can be addressed through a device re-flash [25].
• Device manufacturers and mobile operators must cope with difficult software releases and fix the bug with new software. FOTA provides an easy, efficient, and meaningful mechanism to address software defects before or after device launch and avoid costly recalls [25].

• Study shows that a device recall costs about 40$. This is very high expensive for any device manufacturer and mobile operator. FOTA gives solutions to eliminating this high cost and saving device recall cost [25].

5.7 GOTA METHOD

The abbreviation GOTA spells out as Google Over The Air. This is a firmware software package pushed through Google server. In a certain time period, the device has to check for the update on core Android code. For this update, the device needs to be register on Google server. Google server must upload the software package on their server. End user must have the particular version on their device. The device must be registered by its IMEI (International Equipment Mobile Identity). There are certain criteria by Google to perform GOTA, like each device should get approval from Google, it must pass through Compatible Test Suite(CTS) and Compatibility Definition Document(CDD), Google Mobile Service should be deployed in the device, and the user must sign in to Google account [27]. If update is available then automatically download will start in the background and then the package will be installing as well.

5.8 WEB-UPGRADE METHOD

Web Upgrade is the least used method for device upgrade. In this upgrade, the user just needs to connect the phone to a laptop via USB cable. Each device or carrier company provides a software tool for web upgrade. It is a very easy method to follow. Web Upgrade is particular important when the software package size is big. Because of so many changes in features and for the better performance, much easier user experience, the size of package will be very high (5-6 times than FOTA package). It goes through three short stages like analysis, download and install. This method takes around 10 minutes to complete as per package size and device manufacturer’s requirements. USB should be connected throughout the process.
CHAPTER 6

FUTURE WORK AND CONCLUSION

6.1 FUTURE WORK

Android operating system plays extremely well in smartphone industry. However, there is always a room for improvement. Here I would like to present a few limitations of Android operating system which might be responsible for a bad user experience. Android is an open source operating system, so it also has major security concerns. Android-based smartphones are growing rapidly in numbers. Smartphones contain secure information like online banking access, paypal accounts, secured mail and message services, etc. Software based security vulnerabilities are an extreme consequence of secure information available in the phone. When these vulnerabilities are targeted to retrieve the secure information from the phone, then they are termed as security breaches or attacks. Such attacks had increased in number in the past but their effectiveness had decreased. Now the security attacks are growing more efficient and concentrated, which has increased the rate of favorable outcome [28].

This is one of the major concerns related to Android as a mobile operating system. While multitasking, Android operating system sometimes renders sluggish user interface performance, and blank screen freezes on the device. Compatibility issues are observed with certain applications developed for the Android by vendors. ANR (Application not Responding) and Force Close errors which we have seen in Chapter 4, Call drops also occur with a noticeable frequency due to hand over in the radio interface layer.

The above mentioned concerns for security related enhancements need to be addressed in future versions. Comparisons with other closed source operating systems might help fill this gap and provide a system which could gather all positive features from various sources to make a robust and secure device.
6.2 Conclusion

The Telecom industry is a mercurial entity. The smartphone era started with a loud bang, and it has made the users of formerly used feature phone devices consider switching to this advanced technology. Android started with a 2% market share in 2008, and now it has more than 60% smartphone users. Android provides an extremely user friendly experience with a very supportive ecosystem of applications being continually updated and enhanced by developers from the parent company as well as other extension developers all around the world. There are certain inadequacies which I have come across as a user and researcher, but they could be subjective. There are certain features and enhancements which I would expect to be included in future versions, and new ones might be added as they become available or, rather invented.

My research covers the basics of Android, its functioning as a system, the different upgrade methods, and logging and debugging mechanism. In conclusion, Android’s future is happening now, and further enhancements to this already diverse and multifunctional platform will only take software engineering to the next level.
REFERENCES


