ANDROID BASIC ARCHITECTURE INCLUDING OPERATING SYSTEM USING THEIR APPLICATION

Abu Sarwar Zamani*

Md. Mobin Akhtar**

Ahmad Talha Siddique***

Dr. Safdar Tanweer****

Abstract:

This paper discusses about Android which is software developed for the AndroidTM mobile devices. Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license.

Keywords: Android, Dalvik Virtual Machine, Linux Kernel, Intents, Services, Binder Interfaces

* College of Science and Humanity, Al-Quwaiyyah, Shaqra University, Kingdom of Saudi Arabia.

** Al-Quwaiyyah Community College, Shaqra University, KSA.

*** Noida International University, Noida, India.

**** Jamia Hamdard (Hamdard University), New Delhi, India.

1 Introduction:

Android is a software platform and operating system for mobile devices, based on the Linux kernel, and developed by Google and later the Open Handset Alliance. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries. Applications written in C and other languages can be compiled to ARM native code and run. Google released most of the Android code under the Apache license, a free-software and open source license. Open Handset Alliance, is a consortium of several companies. These companies which aim to develop technologies that will significantly lower the cost of developing and distributing mobile devices and services. The Android platform is the first step in this direction -- a fully integrated mobile "software stack" that consists of an operating system, middleware, user-friendly interface and applications. License Android is under version 2 of the Apache Software License (ASL).

1.1 The Birth of Android

In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA. Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc), Nick Sears (once VP at T-Mobile), and Chris White (one of the first engineers at WebTV). At the time, little was known about the functions of Android Inc. other than they made software for mobile phones. At Google, the team, led by Rubin, developed a Linux-based mobile device OS which they marketed to handset makers and carriers on the premise of providing a flexible, upgradeable system. It was reported that Google had already lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

1.2 Features

1.2.1. Application Framework

It is used to write applications for Android. Unlike other embedded mobile environments, Android applications are all equal, for instance, an applications which come with the phone are

no different than those that any developer writes. The framework is supported by numerous open source libraries such as openssl, SQLite and libc. It is also supported by the Android core libraries. From the point of security, the framework is based on UNIX file system permissions that assure applications have only those abilities that mobile phone owner gave them at install time.

1.2.2. Dalvik Virtual Machine

It is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also tuned to the CPU attributes. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing. Conversion between Java classes and .DEX format is done by included "dx" tool.

1.2.3. Integrated Browser

Google made a right choice on choosing WebKit as open source web browser. They added a two pass layout and frame flattening. Two pass layout loads a page without waiting for blocking elements, such as external CSS or external JavaScript and after a while renders again with all resources downloaded to the device. Frame flattening converts founded frames into single one and loads into the browser. These features increase speed and usability browsing the internet via mobile phone.

1.2.4. Optimized Graphics

As Android has 2D graphics library and 3D graphics based on OpenGL ES 1.0, possibly we will see great applications like Google Earth and spectacular games like Second Life, which come on Linux version. At this moment, the shooting legendary 3D game Doom was presented using Android on the mobile phone.

1.2.5. SQLite

Extremely small (< 500kb) relational database management system, is integrated in Android. It is based on function calls and single file, where all definitions, tables and data are stored. This simple design is more than suitable for a platform such as Android.

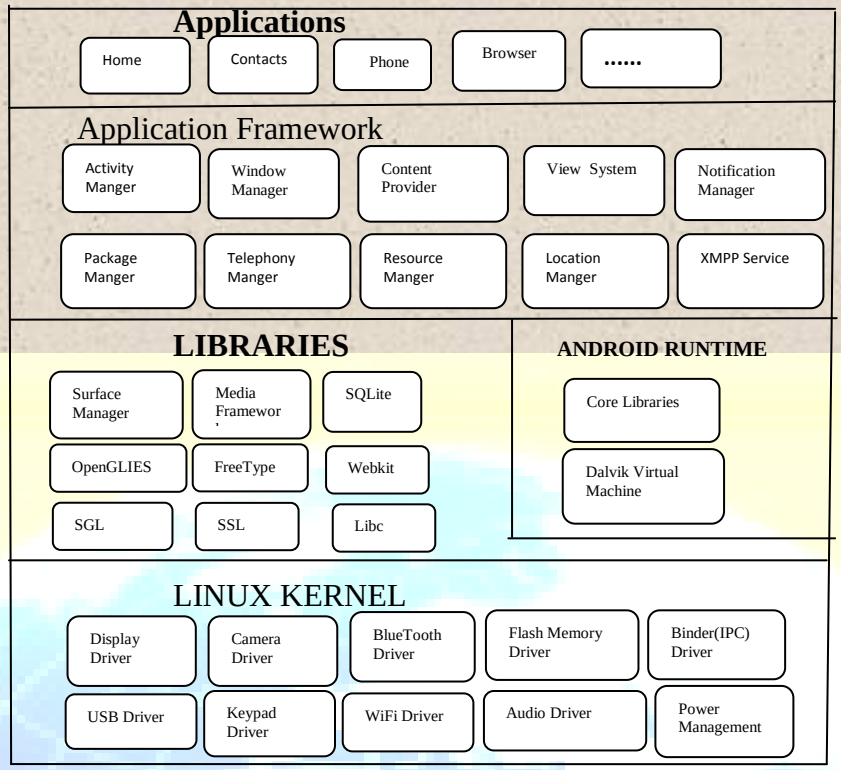
2. Platform:

First and foremost, android is a software stack for mobile devices. This means that high on the list of priorities is the preservation of battery power and the efficient management of limited memory resources.

1. The Acorn RISC Machine (ARM) Linux core forms the solid base upon which all the other layers stand. Linux is a proven technology that is highly reliable, and the ARM processor family is known for high performance on very low power requirements.
2. The libraries provide the reusable and sharable low-level code for basic functions such as codecs — software for coding and decoding digital sound and video — functions for the presentation of rich graphics on a small displays, secure shell support for encrypted TCP/IP traffic into the cloud, as well as component support for Web browsing (WebKit), SQL database functionality (SQLite), and standard C library functionality you would expect in a Linux system.
3. The Dalvik run-time byte-code interpreter, which strongly resembles the Java™ language byte-code interpreter, adds a few distinct features that uniquely define the security and power-preserving model of Android.
4. The Android application framework enables you to use and replace components as you see fit. These high-level Java classes are tightly integrated components that define the Android API.
5. The Android core applications include the WebKit browser, Google calendar, Gmail, Maps application, SMS messenger, and a standard e-mail client, among others. Android applications are written in the Java programming language

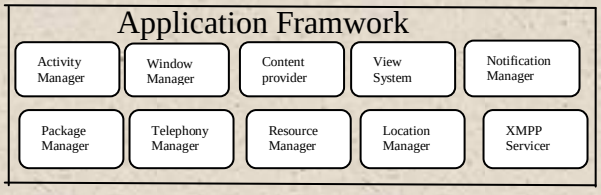
Android is not a single piece of hardware; it's a complete, end-to-end software platform that can be adapted to work on any number of hardware configurations.

2.1. Android Architecture



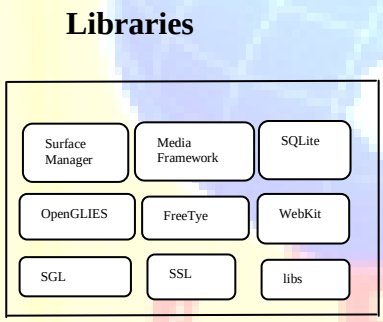
Android uses Linux for its device drivers, memory management, process management, and networking. The next level up contains the Android native libraries. In this layer you can find the Surface Manager (for compositing windows), 2D and 3D graphics, Media codecs (MPEG-4, H.264, MP3, etc.), the SQL database (SQLite), and a native web browser engine (WebKit). Next is the Android runtime, including the Dalvik Virtual Machine. Dalvik runs dex files, which are converted at compile time from standard class and jar files. Dex files are more compact and efficient than class files, an important consideration for the limited memory and battery powered devices that Android targets. The next level up is the Application Framework layer. The most important component of the framework is the Activity Manager, which manages the life cycle of applications and a common “back-stack” for user navigation.

2.1.1 Application Framework



After that, there is Application Framework, written in Java language. It is a toolkit that all applications use, ones which come with mobile device like Contacts or SMS box, or applications written by Google and any Android developer. It has several components. The Activity Manager manages the life circle of the applications and provides a common navigation back stack for applications, which are running in different processes. The Package Manager keeps track of the applications, which are installed in the device. The Windows Manager is Java programming language abstraction on the top of lower level services that are provided by the Surface Manager. The Telephony Manager contains of a set of API necessary for calling applications.

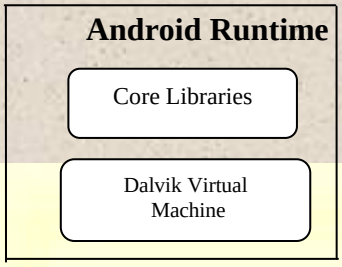
2.1.2. Libraries



In the next level there are a set of native libraries written in C/C++, which are responsible for stable performance of various components. For example, Surface Manager is responsible for composing different drawing surfaces on the mobile screen. It manages the access for different processes to compose 2D and 3D graphic layers. OpenGL ES and SGL make a core of graphic libraries and are used accordingly for 3D and 2D hardware acceleration. Moreover, it is possible to use 2D and 3D graphics in the same application in Android. The media framework was provided by Packet Video, one of the members of OHA. It gives libraries for a playback and recording support for all the major media and static image files. Free Type libraries are used to render all the bitmap and vector fonts. For data storage, Android uses SQLite. As mentioned before, it is extra light rational management system, which locates a single file for all operations

related to database. WebKit, the same browser used by Apples' Safari, was modified by Android in order to fit better in a small size screens.

2.1.3 Android Runtime



Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

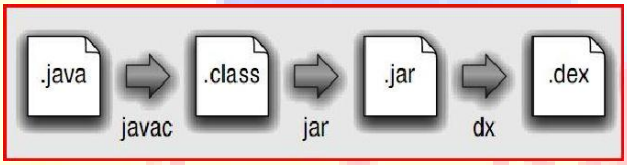
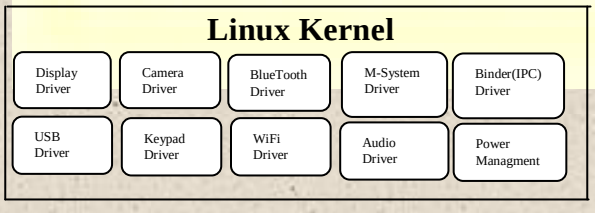


Figure: Conversion from .java to .dex file

2.1.4 Linux Kernel



Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems.

2.2. IDE and Tools

Android SDK

Beside of the actual java class library the Android SDK (latest version 1.1r1) contains all the tools that are necessary to build an Android application.

Developer Tools

As already mentioned above the SDK comes with a bunch of tools that relieve the creation of an Android app. In the following only the most important tools are described:

1. aapt - Android Asset Packaging Tool Creates *.apk-files which contain all the resources as well as the program itself. Those ZIP-format based files can be transferred to and installed on an Android phone or the emulator.
2. adb – Android Debug Bridge The main purpose of this tool is to set up connections to a real Android device or an Android emulator instance in order to transfer and install (apk)-files on it. With adb the developer also has the possibility to remote control the devices shell.
3. dx – Dalvik Cross-Assembler The dx is used for merging and converting Java-Standard-ByteCode Classes (*.class) into one single binary file (*.dex) that can be executed by the Dalvik VM. These *.dex-files are subject to be put into an *.apk-file together with resource files.

4. ddms - Dalvik Debug Monitor Service This tool provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process and radio state information, incoming call and SMS spoofing, location data spoofing, and more.

2.2.1 Application Layer

At the top of Android Architecture we have all the applications, which are used by the final user. By installing different applications, the user can turn his mobile phone into the unique, optimized and smart mobile phone. All applications are written using the Java programming language.

3. Security Issues:

Android is a multi-process system, in which each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data. Security Architecture A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or e-mails), reading or writing another application's files, performing network access, keeping the device awake, etc. An application's process is a secure sandbox. It can't disrupt other applications, except by explicitly declaring the permissions it needs for additional capabilities not provided by the basic sandbox.

Android mobile phone platform is going to be more secure than Apple's iPhone or any other device in the long run. There are several solutions nowadays to protect Google phone from various attacks. One of them is security vendor McAfee, a member of Linux Mobile (LiMo) Foundation. This foundation joins particular companies to develop an open mobile-device software platform. Many of the companies listed in the LiMo Foundation have also become

members of the Open Handset Alliance (OHA). As a result, Linux secure coding practice should successfully be built into the Android development process. However, open platform has its own disadvantages, such as source code vulnerability for black-hat hackers. In parallel with great opportunities for mobile application developers, there is an expectation for exploitation and harm. Stealthy Trojans hidden in animated images, particular viruses passed from friend to friend, used for spying and identity theft, all these threats will be active for a long run. Another solution for such attacks is SMobile Systems mobile package. Security Shield–an integrated application that includes anti-virus, anti-spam, firewall and other mobile protection is up and ready to run on the Android operating system. Currently, the main problem is availability for viruses to pose as an application and do things like dial phone numbers, send text messages or multi-media messages or make connections to the Internet during normal device use. It is possible for somebody to use the GPS feature to track a person's location without their knowledge. Hence SMobile Systems is ready to notify and block these secure alerts. But the truth is that it is not possible to secure your mobile device or personal computer completely, as it connects to the internet. And neither the Android phone nor other devices will prove to be the exception.

3.1 Intents

Intents are an Android-specific mechanism for moving data between Android processes and are at the core of much of Android's IPC. They don't enforce security policy themselves, but are usually the messenger that crosses the actual system security boundaries. To allow their communication role *Intents* can be sent over *Binder* interfaces (since they implement the *Parcelable* interface). Almost all Android IPC is actually implemented through *Binder*, although most of the time this is hidden from us with higher level abstractions

3.2 Services

Services are long running background processes provided by Android to allow for background tasks like music playing or running of a game server. They can be started with an *Intent* and optionally communicated with over a *Binder* interface by calling *Context*'s bindService()

method. *Services* can also be secured by adding a *permission* check to their <service> tag in the AndroidManifest.xml. *Binder* interfaces can check permissions on their caller, allowing them to enforce more than one permission at a time or different permissions on different requests. *Services* therefore provide lots of ways to make sure the caller is trusted, similar to *Activities*, *BroadcastReceivers* and *Binder* interfaces.

Calling a *Service* is slightly trickier. This hardly matters for scheduling MP3s to play, but if you need to make sensitive calls into a *Service*, like storing passwords or private messages, you'll need to validate the *Service* you're connect to is the correct one and not some hostile program32 that shouldn't have access to the information you provide.

3.3 Binder Interfaces

Binder is a kernel device driver that uses Linux's shared memory feature to achieve efficient, secure IPC. System services are published as *Binder* interfaces and the AIDL (Android Interface Definition Language) is used not just to define system interfaces, but to allow developers to create their own *Binder* clients and servers. The terminology can be confusing, but servers generally subclass *android.os.Binder* and implement the onTransact() method while clients receive a binder interface as an *android.os.IBinder* reference and call its transact() method. Both transact() and onTransact() use instances of *android.os.Parcel43* to exchange data efficiently. Android's support for *Binder* includes the interface *Parcelable. Parcelable* objects can be moved between processes through a *Binder*.

3.4 Android Permissions Review

Permissions also have a protection level (called *protectionLevel* as shown above).

1. **Normal:** Permissions for application features whose consequences are minor like VIBRATE which lets applications vibrate the device. Suitable for granting rights not generally of keen interest to users, users can review but may not be explicitly warned.
2. **Dangerous:** Permissions like WRITE_SETTINGS or SEND_SMS are dangerous as they could be used to reconfigure the device or incur tolls. Use this level to mark permissions

users will be interested in or potentially surprised by. Android will warn users about the need for these permissions on install.

3. **Signature:** These permissions can only be granted to other applications signed with the same key as this program. This allows secure coordination without publishing a public interface

4. **Signature or System:** Similar to Signature except that programs on the system20 image also qualify for access. This allows programs on custom Android systems to also get the permission. This protection is to help integrate system builds and won't typically be needed by developers

4. Advantages:

- **Open**- Android allows you to access core mobile device functionality through standard API calls.
- **All applications are equal**- Android does not differentiate between the phone's basic and third-party applications -- even the dialer or home screen can be replaced.
- **Breaking down boundaries**- Combine information from the web with data on the phone -- such as contacts or geographic location -- to create new user experiences.
- **Fast and easy development**- The SDK contains what you need to build and run Android applications, including a true device emulator and advanced debugging tools.

5. Disadvantages:

- **Security**- Making source code available to everyone inevitably invites the attention of black hat hackers.
- **Open Source**- A disadvantage of open-source development is that anyone can scrutinize the source code to find vulnerabilities and write exploits.
- **Login**- Platform doesn't run on an encrypted file system and has a vulnerable log-in.

- **Incompetence** - Google's dependence on hardware and carrier partners puts the final product out of their control.

6. Future Possibilities:

Android sits alongside a new wave of mobile operating systems designed for increasingly powerful mobile hardware. Windows Mobile and Apple's iPhone now provide a richer, simplified development environment for mobile applications. However, unlike Android, they're built on proprietary operating systems that often prioritize native applications over those created by third parties and restrict communication among applications and native phone data. Android offers new possibilities for mobile applications by offering an open development environment built on an open source Linux kernel. Hardware access is available to all applications through a series of API libraries, and application interaction, while carefully controlled, is fully supported.

In Android, all applications have equal standing. Third-party and native Android applications are written using the same APIs and are executed on the same run time. Users can remove and replace any native application with a third-party developer alternative; even the dialer and home screens can be replaced.

- Google Android Sales to Overtake iPhone in 2012
- The OHA is committed to make their vision a reality: to deploy the Android platform for every mobile operator, handset manufacturers and developers to build innovative devices
- Intel doesn't want to lose ownership of the note book market, so they need to prepare for anything, including Android
- Fujitsu launched an initiative to offer consulting and engineering expertise to help run Android on embedded hardware, which aside from cellphones, mobile internet devices, and portable media players, could include GPS devices, thin-client computers and set-top boxes.

Conclusion:

Android applications have their own identity enforced by the system. Applications can communicate with each other using system provided mechanisms like files, *Activities*, *Services*, *BroadcastReceivers*, and *ContentProviders*. If you use one of these mechanisms you need to be sure you are talking to the right entity — you can usually validate it by knowing the permission associated with the right you are exercising. If you are exposing your application for programmatic access by others, make sure you enforce permissions so that unauthorized applications can't get the user's private data or abuse your program. Make your applications security as simple and clear as possible. When communicating with other programs, think clearly about how much you can trust your input, and validate the identity of services you call. Before shipping, think about how you would patch a problem with your application.

This background information should be sufficient for you to be able to understand the presentation Understanding Android application security is the starting point, not the end point for being able to understand Android distribution security.

References:

- Dalvik Virtual Machine specification: available in the Android source code, at dalvik/docs
- Android developers (2011, Mar 03). Android Architecture [Online]. Available:http://developer.android.com/guide/basics/what-is-android.html
- SQLite (2011, Mar 10). About SQLite [Online]. Available: http://www.sqlite.org/about.html
- Chu, E. (2008, August 28). *Android Market: a user-driven content distribution system*. Retrieved August 30, 2008, from Android Developer's Blog: http://android-developers.blogspot.com/2008/08/android-market-user-driven-content.html
- Google Inc. (2008, August 29). *Security and Permissions in Android*. Retrieved August 30, 2008, from Android - An Open Handset Alliance Project: http://code.google.com/android/devel/security.html
- Burns, Jesse (2009, October) DEVELOPING SECURE MOBILE APPLICATIONS FOR ANDROID. http://www.isecpartners.com/files/iSEC_Securing_Android_Apps.pdf