ANDROID OPERATING SYSTEM

Abstract

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. Open - Android allows 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 need to build and run Android applications, including a true device emulator and advanced debugging tools.

.

1. Introduction

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. Open - Android allows 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 need to build and run Android applications, including a true device emulator and advanced debugging tool

1.1 **The birth of Android**

Google Acquires Android Inc.

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.

1.2 Open Handset Alliance Founded

On 5 November 2007, the Open Handset Alliance, a consortium of several companies which include Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel and NVIDIA, was unveiled with the goal to develop open standards for mobile devices. Along with the formation of the Open Handset Alliance, the OHA also unveiled their first product, Android, an open source mobile device platform based on the Linux operating system

1.3 Hardware

Google has unveiled at least three prototypes for Android, at the Mobile World Congress on February 12, 2008. One prototype at the ARM booth displayed several basic Google applications. A'd-pad' control zooming of items in the dock with a relatively quick response

2.Features of Android

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

3.Android Architecture

Fig.1 Android Architecture

3.1 Application Framework

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, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and lat files
- A Notification Manager that enables all applications to display custom alerts in the status bar
- An Activity Manager that manages the life cycle of applications and provides a common navigation back stack

3.2 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on Packet Video's Open

CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG

- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **Free Type** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

3.3 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.

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool “dx”, which converts generated byte code from .jar to .dex file, after this byte code becomes much more efficient to run on the small processors.

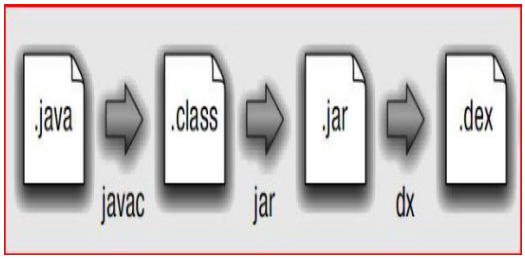

Figure2: Conversion from .java to .dex file

As the result, it is possible to have multiple instances of Dalvik virtual machine running on the single device at the same time. The Core libraries are written in Java language and contains of the collection classes, the utilities, IO and other tools.

3.4 Linux Kernel

Android Architecture is based on Linux 2.6 kernel. 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. Android provides the support for the Qualcomm MSM7K chipset family. For instance, the current kernel tree supports Qualcomm MSM 7200A chipsets, but in the second half of 2008 we should see mobile devices with stable version Qualcomm MSM 7200, which includes major features:

- WCDMA/HSUPA and EGPRS network support
- Bluetooth 1.2 and Wi-Fi support
- Digital audio support for mp3 and other formats
- Support for Linux and other third-party operating systems
- Java hardware acceleration and support for Java applications
- Qcamera up to 6.0 megapixels
- gpsOne – solution for GPS

4. Architecture for secure data storage

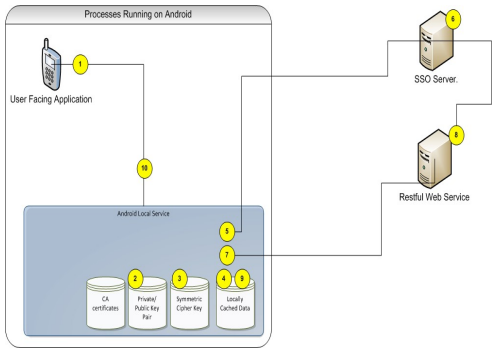

Figure 3 Android and Secure Local Data Storage

Secure data storage solution that could potentially be deployed on Android. It is as shown in the figure. However, many shortcomings of the design have been addressed. Additional security highlights will be presented at the end of the section.

Using figure 3, we have the following workflow:

- The user enters his credentials on the handset.
- The credentials are not sent to the SSO service over the network. Instead, the credentials are used as the passphrase to decrypt the local public/private key pair of the user. We define the public/private key pair to be of type RSA and of at least 4096 bits in size. Already we gain the advantage that the user's password is not sent over the network.
- The private key is used to decrypt the symmetric cipher key. The symmetric cipher key is used to

encrypt/decrypt any locally cached data. A strong symmetric cipher like 3DES is used

- All data found in the local cache is encrypted with the symmetric cipher key defined in step #3.
- If the requested data is not locally cached or expired. We must communicate with the SSO service again to be able to receive fresh data from the Restful web services. However, unlike the architecture presented in section 2 of this document, we login to the SSO server using a hostile challenge based on the private key of the user. As such, we login with the
- infrastructure is once again used to setup a secure communication

- web services are procured from the same certificate authority that shipped with the phone.
- On reception of a request, the SSO token is extracted from the request. The web service calls upon the SSO system to authorize the operation.
- . On reception of the data, the symmetric cipher described in bullet #3 above is used to encrypt the data before it reaches any local persistent storehouse
- Data is returned to the user facing application

5. Execution environment

SSO system using public/private key infrastructure. The user name and the password are never sent over the network. The SSO system can identify the user based on this challenge and returns a 496 bit alpha numeric token.

- The tokens generated by the SSO system are set to automatically expire after a given period of time
- On reception of the SSO token. The Android background application can now communicate with any Restful web services that adhere to the same SSO federation. Public/private key

channel between the phone and the server. The certificates of the servers that host the

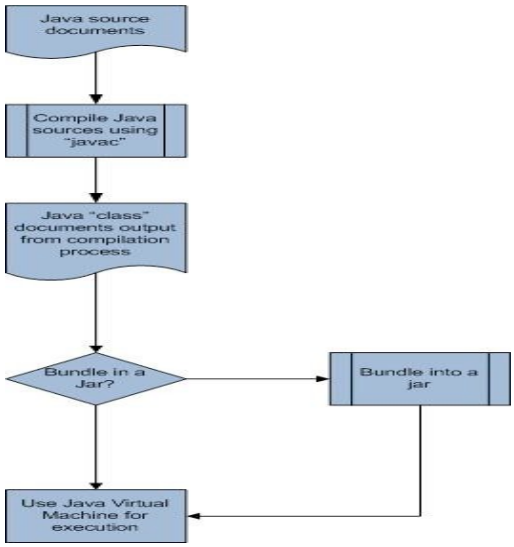

Figure 4. Regular Java Execution Process

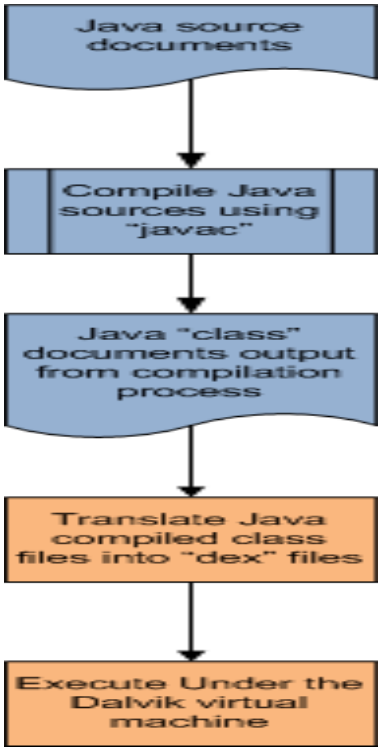

Figure 5 .Android Execution Environment

Figures 4 and 5 represent the regular Java and Android execution paths respectively. It is interesting to note here however is that the Android compilers do not operate on Java
language code. Instead, the Android translators work on the resulting Java bytecode emitted from a traditional Java compiler.

As such, it is possible to reuse existing Java libraries, even if the original source code is not available. Such libraries must meet stringent requirements however, they need to:

1. Adhere to the Java SE 5 dialect
2. Not use any Java classes or packages found in Java SE 5 not found in the Android platform
3. Not use any packages or classes specific to the Sun Microsystems platform
4. Still behave in a predictable manner under the Apache Harmony Java environment

Following these guidelines, it's possible to integrate existing Java source code, packages and libraries piecemeal. Special care will be needed in the integration phase of such code but the potential savings offered by such integration far outweighs the cost of rewriting well-coded, well-documented and well-tested libraries ready for use. Furthermore, it is expected that has Apache Harmony matures, more and more compatibility issues will be resolved further increasing the pool of available Java code that will be able to execute unmodified under the Android platform.

6. The Dalvik Virtual Machine

The Dalvik virtual machine is an interpreter only machine optimized for use on low powered, low memory devices like phones. Notably, Dalvik does not make use of just in time (JIT) Compilation to improve the performance of an application at runtime. Furthermore, Dalvik is not a Java virtual machine. This is because Dalvik is unable to read Java bytecode34, instead it uses its own byte code format called "dex". Google claims this format allows battery power to be better-conserved at all different stages of execution of an application. This means that standard Java SE applications and libraries cannot be used directly on the Android Dalvik virtual machine.

Dalvik however stands at the center of the Android value proposition. Its low electrical power consumption, rich libraries, and unified, non-fragmented application programming interfaces make it stand out, or so Google hopes, over the fragmented ecosystem that is Java ME35 today.

Furthermore, since Dalvik uses the Java programming language but not the Java execution environment (JVM), Google is free to develop Android without the need to license or obtain certification from Sun Microsystems Inc, the legal owner of the Java trademark and brands.

7. Conclusion

Android is a truly open, free development platform based on Linux and open source. Handset makers can use and customize the platform without paying a royalty.

A component-based architecture inspired by Internet mash-ups. Parts of one application can be used in another in ways not originally envisioned by the developer. can even replace built-in components with own improved versions. This will unleash a new round of creativity in the mobile space.

1. Android is open to all: industry, developers and users

2. Participating in many of the successful open source projects

3. Aims to be as easy to build for as the web.

4. Google Android is stepping into the next level of Mobile Internet

8. References

[1] White paper for "A Spectrum White Paper:Thoughts on Google Android" from Spectrum data Technology.http://www.spectrumdt.com

[2] http://code.google.com/android/ - Google Android official webpage.

[3]http://www.openhandsetalliance.com/ - Open Handset Alliance webpage

[4]http://en.wikipedia.org/wiki/Android_(mobile_phone_platform)– Wikipedia information

[5] http://googleblog.blogspot.com/ - Official Google Blog