IJCAT - International Journal of Computing and Technology, Volume 3, Issue 2, February 2016
ISSN : 2348 - 6090
www.IJCAT.org

Android OS Architecture and Services

[1] Ranjit R. Keole, [2] Akshata N. Rathod

[1] Professor (Dept. of Information Technology)
HVPM COET,
Amravati, (M.S.), India

[2] First Year M.E.(CS&IT)
HVPM COET,
Amravati (M.S.), India

Abstract – Mobile devices have seen an extensive amount of development in recent years, but one question is still looming and nobody seems to have the answer: what is 'standard' for the mobile platform?Many companies have already written their own in-house operating systems for the devices they manufacture such as Symbian or iPhone OS. However, with the existence of so many closed-source operating systems, no rational company would want to disclose their secrets and lose their edge on the competition. This presents a problem where software developers can't write their code to be generalized. The Android team hopes to solve this on two levels. Firstly, it seeks to arrive at a common open-source operating system that any mobile device can run on. Secondly, it seeks to make developing applications for these mobile phones more general and hardware-agnostic.

Keywords - **Dalvik VM, Linux, Sandbox.**

1. Introduction

Android is a software platform and operating system for mobile devices, based on the Linux kernel, 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, but this development path is not officially supported by Google. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 48 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache license, a free-software and open source license.

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 cofounders 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.

The Android Platform:
Android is an operating system and a software platform upon which applications are developed. A core set of applications for everyday tasks, such as Web browsing and email, are included on Android handsets. As a product of the Open Handset Alliance's vision for a robust and open source development environment for wireless, Android is an emerging mobile development platform. The platform was designed for the sole purpose of encouraging a free and open market that all mobile applications phone users might want to have and software developers might want to develop.

Android Platform Differences:
Android is hailed as "the first complete, open, and free mobile platform."

Complete:
The designers took a comprehensive approach when they developed the Android platform. They began with a secure operating system and built a robust software framework on top that allows for rich application development opportunities.

Open:
The Android platform is provided through open source licensing. Developers have unprecedented access to the handset features when developing Applications.

Free:
Android applications are free to develop. There are no licensing or royal fees to develop on the platform. No required membership fees. No required testing fees. No required signing or certification fees. Android applications can be distributed and commercialized in a variety of ways.

Android: A Next Generation Platform
Although Android has many innovative features not available in existing mobile platforms, its designers also leveraged many tried-and-true approaches proven to work in the wireless world. It's true that many of these features appear in existing proprietary platforms, but Android combines them in a free and open fashion, while simultaneously addressing many of the flaws on these competing platforms.

Android is the first in a new generation of mobile platforms, giving its platform developers a distinct edge on the competition. Android's designers examined the benefits and drawbacks of existing platforms and then incorporate their most successful features. At the same time, Android's designers avoided the mistakes others suffered in the past.

Open Handset Alliance:
Open Handset Alliance, is a consortium of several companies which include Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel and NVIDIA, ... 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.

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.

License:
Android is under version 2 of the Apache Software License (ASL). The Apache license allows manufacturers and mobile operators to innovate using the platform without the requirement to contribute those innovations back to the open source community.

Hardware 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. There are five distinct layers to the Android system stack:

- 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.

- 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.

- 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. Every application currently running, for example, has its own user ID and its own copy of the interpreter running to strictly separate processes for security and reliability.

- 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.

- 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, and you can download many more from the Android market on the fly.

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. Everything is there, from the boot loader all the way up to the applications. And with an Android device already on the market, it has proven that it has what it takes to truly compete in the mobile arena.

Executive Summary:

This paper attempts to study the present conditions of Android OS and unveils the predicted future market possibilities for Android, based on results from several research firms, using current market statistics and popularity among developers and end-users. All the flimflams and excitement about the costlier iphones and Blackberrys are vanishing, after the arrival of the most anticipated, open source mobile operating system, the Google Android, which is fated to turn the industry upside down. Despite the growth and popularity for iPhones and Blackberrys, it is predicted that, Android will make a history in sales and on acquiring the market share, slicing down the markets of both Symbians and iPhones.

This paper will elaborately examine the predictions about the future of Android phones, considering the present facts and reasons.

The Android Tal:

Open Handset Alliance (OHA) a confederation of 50 Telecoms, mobile hardware, and software companies, headed by Google, was found on 5th of November, 2007. The consortium's goal is deploy, the advanced open standards for mobile devices. Android is an open source mobile OS platform, purely based on the Linux operating system, Apache harmony, and Dalvik Virtual machine and was first developed by Google, later backed by the Open Handset Alliance.

A simple and attractive thing about Android, is its Java-like language based on Google-developed Java libraries. Recently, for the first time, Google released the Native Development Kit (NDK) for Android which enables programmers to develop programs and native application that could run on the device.

Android: Breaking the "Walled Garden"

Like Apple Appstore, Google opened its Android market, allowing the apps developers to publish their apps without any restrictions. Unlike Apple's Appstore, Google Android market will not have any restrictions for third party development and will not run an apps approval systems. And Android will be breaking another 'Walled garden', that's the mobile carrier support. In US, AT&T had acquired the rights to sell Apple's iPhones for the next five years from the date of its release. And in case of the Blackberrys, it is not a fully carrier-independent handset, since the major part of the sale happens through its different carriers, worldwide.

This approach had left people frustrated, on sticking to a monopolistic mobile carrier, irrespective of their wish to select a different carrier. Since, Android is a open source operating system, it could leverage the advantages of device-independency and service provider-independency.

What's so different in Android:

The good news is for both the consumers and developers. While consumers could enjoy a low-cost Smart phones running Android, developers were given an unrestricted customization rights. From a developer's point of view, Android has several advantages, as listed below:

The entire Application framework can be reused and replaced by selective components. Dalvik virtual machine enhances the power management systems (Learn about Dalvik VM in the following subtitle)

Support for 2D and 3D graphics (OpenGL ES 1.0), So lot of business for animation developers. Reliable and enhanced data storage (using SQLite framework)

Developers can create media common applications since it supports common media file formats(MPEG, MPEG3, MPEG4, H.286, AAC, AMR, JPG, PNG, GIF and more)
GSM, EDGE, 3G, HSCSD, Wi-Fi network applications support (Depends on hardware)

Open source Web-Kit Engine-based web-browser GPS, Navigational compass, Touch-Unlock, and accelerometer applications support (Depends on hardware)

- Androids development environment includes a device emulator, debugger, performance profiling tool, and an Eclipse IDE plug-in

Android: A promising haven for app developers and OEMs

Application development companies, equipment manufacturers, and individual app developers consider, Android platform as the most promising platform due do the cost efficiency in production values. Google has given the opportunity to develop equal native applications, with

which a user can replace the Google bundle with his own non-Google bundle applications. Undoubtedly, Android platform is the true open source platform, but it too has got some limitations. Google tries to hold the platform development by the third party developers. by restricting them to develop android applications using, none other than Dalvik Virtual Machine, while ironically, the major part of the Android is written in C and C++. When Android was introduced, the openness of the platform was hyped as the main strength. Google officials, stated that the Android developers are allowed to furnish whatever they saw fit to run.

This made the entire mobile industry and software development companies to dream about the fruit of the eternal development cycle and the revenue generated through it. Now, number of Androids have began to appear. But the first Android phone G1, released by USbasedT-Mobile, which was a completely packed Google phone. While recently, HTC, who is the manufacturer of the G1 handset, is offering the its own Android-based HTC Magic, ripping off the Google-based bundles and includes other features which is not seen in the other basic Android handsets. This clearly reveals that manufacturers, software developers and phone carriers want to stand different from the global competition on Android platform and customization business. So its obvious, that Android is going to grow like Linux does, offering developers, a chance to develop applications for different versions of Android by different OEMs.

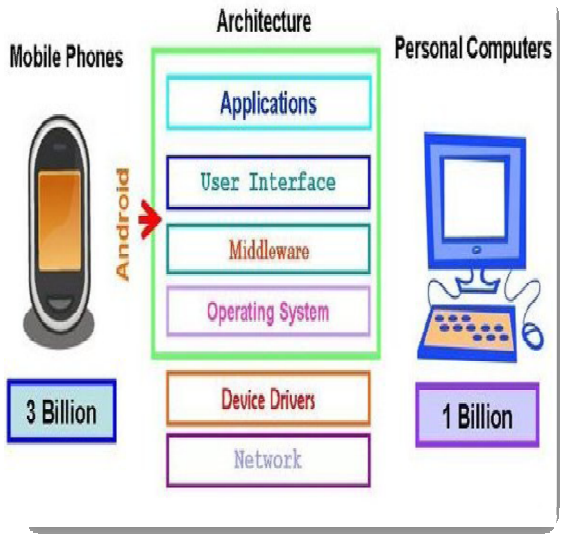

Fig: 1 app developers and OEMs

Positively, Android will began sporting multiple interfaces, which will be modified by different software vendors. Some years back, the same scenario was witnessed in the Windows Mobile world, and that was to scale the awareness, a consumer has of the Windows Mobile. By 2012, Android will be completely customized (in fact, that's already happening), and it will be definitely lucrative for app developers, while the competition will heat up, on which handset or whose carrier's phone, its going to be.

Market Predictions:

Very few but strong predictions about Android are spreading, worldwide. Firstly, Android is going to be bigger in terms of consumer reach than its rival, the iPhone OS. Its just because of a true fact that it will be developed and marketed by all the 50 members of OHA, which includes companies like Google, Samsung, HTC, Sony Ericsson, T-Mobile, Motorola, Vodafone, Sprint, China mobile and other world leaders in telecommunication industry. Secondly, Google's support will make everything possible in this Internet era, but up to now, they hadn't started making money from their Android-based activities. On the other hand, by 2012, apart from Symbian and Android, iPhone will target its businesses development towards its rival, the Blackberry in their segment. But predictions say that, Android is completely made for mass market, and its lack of business features (Unlike, Winslow Mobile and Blackberry, while both them has business exchange compatibility and PC-Synchronization features) will create new chances for its rivals.

According to Gartner, Android's smartphone market share will grow to 14 percent from less than 2 percent by 2012, and the Symbian's slide will continue, giving way to Android. That kind of historical performance by Android would mirror the Apple iPhone's rule. The first generation iPhone was launched in 2007, in the U.S. and immediately, it took the world by storm. On its release, the iPhones grabbed a good 11% of the smartphone market share in the first quarter of 2009 and continued to expand day by day. But the predictions about Android's gathering momentum, will overtake Apple in just 2 years. Almost all handset vendors are trying to board the Android's bandwagon, while the Google has effectively grown a massive android developer's community, and also, Android is backed by the America's largest mobile network carrier, the T-Mobile. However, the Apple's iPhone remains the overwhelming choice for global users, skyrocketing the sales,

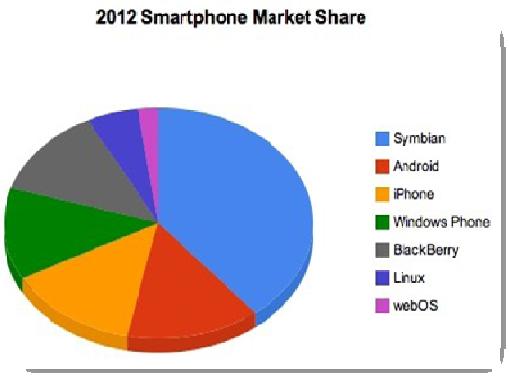

Fig: 2 Graph of 2012 Smartphone Market Share

Fig: 2 Graph of 2012 Smartphone Market Share and are offered by numerous operators around the world. While Apple has successfully grabbing the market inch by inch, by offering a user-friendly, 3G and High speed Internet-capable handset, Android is still fighting in that part, not only benefiting the Apples, but also benefits the RIM's Blackberry, Windows, Palm OS and others. Presently, Android may appear to be an invincible giant, but it will take its own time grab the market in these early times of the super phone-era.

Fig: 3 Table of Platform Market Share

Recently, Google has fore-casted that nearly 20 Android phones, would be released by the members of OHA, before the end of this 2009. It is an amazing progress for a very young open source platform, which is like an 18 month baby fighting with giants. Googles says that the credits goes to the openness of the environment and it feels that the Android is not just an Mobile operating system, but it is a completely Open software development environment for mobile phones.

Android's entry seems very successful, which is driven by the worldwide acceptance, and the thirst for an open source mobile environment backed by countless application development companies and telecommunication leaders. In fact, that every other Mobile OS vendors had identified Android as an acute and critical threat to their future.

Final Comments:
Giants like Nokia and Microsoft are not the part of the OHA, so Android won't become so powerful for now. It depends on the stability and continuous support for the platform.

- Microsoft's 'windows mobile' has a big selling point in the form of , its integrity with Microsoft Office and other Microsoft-owned tools, So, Android has to concentrate on developing PC compatibility apps using the Google Office Apps.

- We are not talking about the very success of Android, since it has its own challenges, like its development task. Though, Android is a complete mobile handset platform, encompassing a mobile operating system, a browser , some middle ware, and other application environment, that all depends upon the future investments, and innovations upon the development of an allencom passing new technology.

Apart from Google, Apps developers and vendors are predicted to be the major beneficiaries, since they would make most out of the Android with its wider market structure backed by all the members of OHA, developing different versions of Android, and driving the Android's apps market to a new edge, defeating all the others.

Features of Android:

Handset layouts: Android can adapt to traditional smart phone layouts, as well other VGA, 2D, and 3D graphics libraries.

Storage: Android uses SQLite to store all its junk-- I mean, information.

Connectivity: Android supports a wide variety of technologies, including Bluetooth, WiFi, GSM/EDGE, and EV-DO.

Messaging: MMS and SMS are available for Android, as well as threaded text essaging. So you can send as many texties as you like.

Web Browser: Android comes pre-loaded with the Web Kit application. Remember, if you don't like it, you can

always switch it out for something else later on thanks to the open source nature of the Google Android backend.

Additional Hardware Support: Got a touch screen you want to put to its full use? No problem. Android is capable of utilizing outside hardware like GPS, accelerometers, and all that other fun stuff.

Java Virtual Machine: Software you write in Java can be compiled in Dalvik Byte codes (say that five times fast. I keep ending up with "Danish light bulb".) These can then be put into a Dalvik Virtual Machine. Basically more robust applications are supported than on some other Mobile Operating Systems.

2. Android OS Architecture

Android can be subdivided into four main layers: the kernel, libraries, applications framework, and applications. As previously mentioned the kernel is Linux. The libraries that come with Android provide much of the graphics, data storage, and media capabilities. Embedded within the libraries layer is the Android runtime which contains the Dalvik virtual machine, which powers the applications. The applications framework is the API that all applications will use to access the lowest level of the architecture.

The Kernel Layer:
As previously mentioned the kernel layer is Linux. Linux was chosen since it has a proven track record in desktop systems and in many cases doesn't require drivers to be rewritten. Linux provides such things as virtual memory, networking, drivers, and power management. Upon examining the kernel shipped with the Android source code, there are not any significant changes to the core functions of the kernel.

Native Libraries Layer:
The native libraries layer provides Android with the capabilities for its core features. Androidis shipped with SGL which acts as the primary 2D graphics renderer. Its counterpart is OpenGL ES which provides 3D graphics support. Android comes packaged with SQLite which takes care of most data storage. The WebKit web rendering engine is also shipped with Android and has been tailored to render web pages for smaller screen sizes. Of particular interest is the Dalvik virtual machine which is a part of this layer.

The Dalvik virtual machine is a bytecode interpreter which is highly optimized for executing on the mobile platform. The bytecodes are converted Java binaries that are very quick and efficient to run on smaller processors. The core libraries are written in Java and provide much of the core classes which would normally be available in a Java virtual machine.

Applications Framework Layer:
This layer and the layer above it are written completely in Java. The applications framework provides all of the major APIs that the applications will use including things like sharing data, accessing the telephony system, and receiving notifications. An important thing to note about Android OS is that all applications use this same framework no matter the author of the application. This is quite a departure from what many other mobile OS designers have chosen to do. For instance the iPhone most certainly differentiates between Apple software and third-party software down to the copy-and-paste feature.

Applications Layer:
All of Android's software is written in Java, which is interpreted by the Dalvik virtual machine. Even the most core features such as the phone and the contacts application reside in this layer. This layer contains software written by the Android team as well as any third-party software that is installed on the device. An effect of allowing third-party developers access to this layer is that the user interface can be overhauled comparatively easily. Third party applications can handle any event that the Android team's application could see (such as the phone ringing).

This means that so long as there is a replacement application for the dialer application, anyone could potentially write their own. Given this model we might expect that, as Android becomes more robust, the user will be able to specify what applications should handle which events.

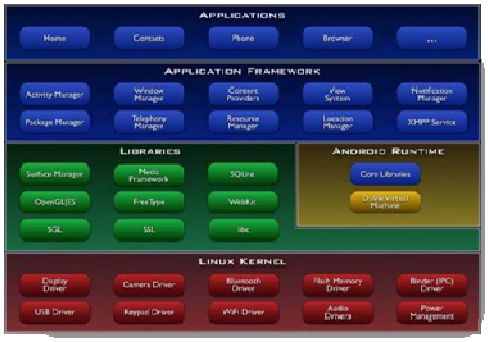

Fig.4 Android Architecture

Operating System(s) :

Android uses Linux for its device drivers, memory management, process management, and networking. However you will never be programming to this layer directly. The next level up contains the Android native libraries. They are all written in C/C++ internally, but you'll be calling them through Java interfaces. 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 core Java libraries are also part of the Android runtime. They are written in Java, as is everything above this layer. Here, Android provides a substantial subset of the Java 5 Standard Edition packages, including Collections, I/O, and so forth. The next level up is the Application Framework layer. Parts of this toolkit are provided by Google, and parts are extensions or services that you write.

The most important component of Android the framework is the Activity Manager, which manages the life cycle of applications and a common "back-stack" for user navigation.

Finally, the top layer is the Applications layer. Most of your code will live here, along side built-in applications such as the Phone and Web Browser.

Security:
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. These permissions it requests can be handled by the operating in various ways, typically by automatically allowing or disallowing based on certificates or by prompting the user. The permissions required by an application are declared statically in that application, so they can be known up-front at install time and will not change after that.

Performance :

Devices hosting Android applications have limited capabilities. That's why code should be efficient, avoid all unnecessary memory allocations, method calls (it takes a lot of time) and so on. In order to make our applications working fast on a mobile device we need to leave back some habits, good from OOP point of view. In a mobile device we are not able to make a full model of reality what we want to operate on. Few things to remember:

Avoid object instantiation: create objects only if it is really necessary, because it costs time and memory. More instances means more-frequent garbage collection what lowers user-experience (freezes).

Use native built-in methods: they're written in C/C++ what makes them faster about 10-100 times than implemented JAVA code (loops etc.). However note that calling native method is more expensive then calling implemented one.

Virtual over interface: in conventional programming it is usual to declare variables as interfaces, i.e.: Map myMap1 = new HashMap(); It is not good for embedded applications. Calling a method from interfaces takes 2 times more time than in normal way: HashMap myMap2 = new HashMap();

Static over virtual: declare methods static if they do not need access to the object's fields. It can be called faster, because it doesn't require a virtual method table indirection. It's also good practice, because you can tell from the method signature that calling the method can't alter the object's state.

Cache field lookups: because accessing object fields is lower than local variables. The same situation is with

methods - i.e. by for-statements, you should cache size() method if it is possible:

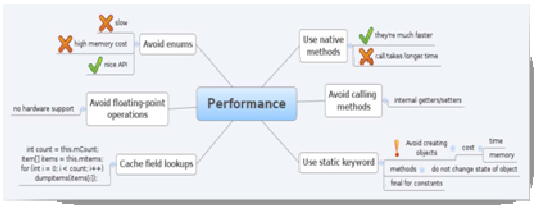

Fig: 5 Performance of Android OS

3. Advantages

There are a host of advantages that Google's Android will derive from being an open source software. Some of the advantages include:

- The ability for anyone to customize the Google Android platform will open up the applications playing field to small and new players who lack the financial muscle to negotiate with wireless carriers like AT&T and Orange.
- The consumer will benefit from having a wide range of mobile applications to choose from since the monopoly will be broken by Google Android.
- Although this will depend on the carrier, one will be able to customize a mobile phones using Google Android platform like never before, right down to the screen.
- Features like weather details, opening screen, live RSS feeds and even the icons on the opening screen will be able to be customized.
- In addition, as a result of many mobile phones carrying Google Android, companies will come up with such innovative products like the location – aware services that will provide users with any information they might be in need of.
- This information could include knowing the location of a nearby convenience store or filling station. In addition the entertainment functionalities will be taken a notch higher by Google Android being able to offer online real time multiplayer games.

4. Limitations

- **Bluetooth limitations**

Google Talk functions and only the simplest implementation of Bluetooth. It'll work with Bluetooth headsets but that's about it; no Bluetooth stereo, no contacts exchange, no modem pairing and no using wireless keyboards. Android uses a non-standard jvm: there is no guarantee that the same software will run on multiple devices

- **Firefox Mobile isn't coming to Android because of Android Limitations**

Fennec won't play nice with Android Market because apps in Android Market need to be programmed with a custom form of Java to run on Android. Mozilla and the Fennec peeps won't have that and won't be releasing any form of Firefox until Google amends the limitation of Android Apps.

Android system services:
Android provides the services expected in a modern operating system such as virtual memory, multiprogramming, and threads, all on a mobile platform. Many of Android's services are a result of including the Linux kernel. However the Android team has added the telephony stack in return.

The Linux CPU scheduling algorithm:
Linux employs a number of different methods for scheduling its processes and its algorithm is very nontraditional in the big picture. There are three scheduling schemes for Linux. Each process is assigned a scheduling scheme depending on the type of task it presents. Real time tasks will often run in the SCHED FIFO or SCHED RR scheme. All other tasks will run in SCHED NORMAL [?]. SCHED NORMAL tasks are handled by a special algorithm and are preempted by SCHED RR and SCHED FIFO tasks. In a few words the algorithm behaves like a hybrid priority queue that rewards process which are not CPU-greedy.

CPU time is divided up into epochs, which are equal slices of time in which the processes can run and use up some of their allotted time, or timeslice. At the end of every epoch, each process' remaining timeslice is halved. More time is added proportional to the processes' nice value, a value specified by the user or by the system's default nice value. The interesting rollover time has a rewarding effect to I/O bound processes. Since I/O bound processes will likely spend a lot of time waiting, the

scheduling algorithm rewards it by letting it keep some of its timeslice.

This way processes which are likely being used by the user are very responsive and will often preempt more non-interactive, CPU-bound processes like batch commands. SCHED RR tasks will preemptSCHED NORMAL tasks and are preempted by SCHED FIFO tasks. These tasks are scheduled according to round-robin rules within their own priority bracket. SCHED FIFO tasks behave quite like the name would suggest. Tasks in this queue are scheduled by priority and arrival time, will preempt any other processes trying to run, and will execute for as long as they please.

Android and file systems:
Android makes use of a multiplicity of file system types, namely due to the expectation of external memory with unsure file system types. As a result Android relies on the standard Linux package to provide for file systems such as ext2 and ext3, vfat, and ntfs. Android itself uses the yaffs2 file system, which is not a part of the standard Linux kernel, as its primary file system. YAFFS is a file system optimized for NAND and NOR flash memory. At the time when it was developed, file systems did exist for flash memory but most of them catered toward chips which were small enough to use small block sizes. This was unsuitable for large NAND flash chips.

YAFFS attempts to solve this by abstracting storage to ”chunks” which scale according to the page size. To scale the method up for considerably large NAND devices, YAFFS has a tweak in the way that it addresses pages. For instance, we might use a page address size of 216 and we may have have 218 chunks to address. We do not have sufficient capabilities to address pages individually, but we can address groups of four pages and search for the desired page from there. The useful thing about this is that now we have the option of neglecting to use RAM at all to augment our file system. We can pretty easily used some sort of indexed referencing scheme to build a file from a string of chunk IDs, and any scanning for particular pages within that chunk will be a limited set of a size equal to the number of chunks divided by the address size. ($218 \div 216 = 4$). This linear probing may seem like a bad idea. In practice, the inefficiency is too small to notice on such a small set of chunks [?]. YAFFS is preferable as a file system in Android since it optimizes the use of NAND devices as storage and also has great efficiency in memory usage.

The radio interface layer:
Consistent with the rest of the design of Android, the Android team has created a way to abstract a phone call. Since the way that handset manufacturers implement the radio device on cellular devices will inevitably vary, Android has to remain ignorant to the way software interacts with hardware to place the call. The Android team solves this problem by creating the Radio Interface Layer Daemon (RILD), which is the way that the applications framework interfaces with the shared libraries.

The RILD’s main function is to provide event-driven middle ground between the applications framework and the radio drivers. The RILD is part of the Radio Interface Layer (RIL), which consists of two major parts: the Android RIL and the Vendor RIL. The Android RIL includes the applications framework which makes the request to the RILD to place a phone call, while the Vendor RIL is the part that is up to the vendor to implement [?]. The Vendor RIL includes the driver that the vendor ships for the hardware implementation of the radio antenna. This model allows any hardware to be implemented for placing phone calls so long as the vendor writes drivers which follow the model.

4. Conclusions

Android is a truly open, free development platform based on Linux andopen source. Handset makers can use and customize the platform without paying aroyalty. A component-based architecture inspired by Internet mash-ups. Parts of oneapplication 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.

References

[1] Ben Elgin, “Google buys android for its mobile arsenal,”http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm, August 2005.

[2] John Cox, “Why google’s gphone won’t kill apple’s iphone,” http://www.networkworld.com/news/2007/100807-google-gphone-iphone.html, October 2007.

[3] Mike Cleron, "Androidology: Architecture overview," http://www.youtube.com/watch?v= Mm6Ju0xhUW8, November 2007.

[4] Josh Aas, "Understanding the linux 2.6.8.1 cpu scheduler," Retrieved from http://www.angelfire.com/folk/citeseer/linux_scheduler.pdf, February 2005.

[5] C harles Manning and Wookey, YAFFS Specifications, Aleph One, Bottisham, UK, version 0.3edition, February 2002, Retrieved from http://www.yaffs.net/yaffs-spec.

[6] Google, Mountain View, CA, Radio Layer Interface, March 2009, See path '/development/pdk/docs/telephony.html' in Android source tree.