Android vs iPhone

Junyao Zhang

April 12, 2010

This is a complete analysis and comparison between Android and iPhone OS. The rest of this report is organized as follows. Section **??** outlines the system architecture, history and detail management configuration. Section **??** discusses the iPhone system. In Section **??**, a comparison between these two systems is presented.

1 Android

Android, originally meaning "robot", is a mobile operating system using a modified version of the Linux kernel. It was initially developed by Android Inc., a firm later purchased by Google,[**?**]and lately by the Open Handset Alliance[**?**]. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries.[8] It empolys the software stack architecture, as shown in Figure 1.

- 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 is implemented by programming language C.
- The middleware level includes *Runtime and Libraries*. The *Runtime* includes *core libraries,* providing most of the functionality available in the core libraries of the Java programming language, and *Dalvik virtual machine* which allows every Android application runs in its own process. The *Libraries* is used by various components of the Android system, such as Media Libraries, 3D libraries, and etc.
- The upper level is *Application framework* and *Application*. *Application framework* is offering developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. All *applications* are written using the Java programming language.

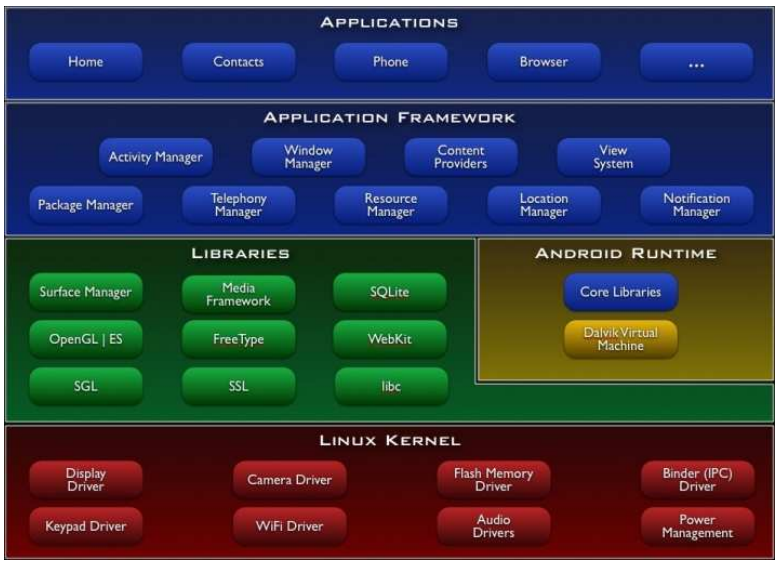

Figure 1: Android System Architecture

1.1 A brief review of the history of Android

In July 2005, Android, Inc., a small startup company based in Palo Alto, California, USA, was bought by Google. At that time Android, Inc. is not well-known execpt that they made software for mobile phones. At Google, a team was set up to produce a mobile device platform that aims to provide a flexible and upgradable system. It is 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[**?**, **?**, **?**]. More speculation that Google would be entering the mobile-phone market came in December 2006[**?**].

In September 2007, InformationWeek covered an Evalueserve study reporting that Google had filed several patent applications in the area of mobile telephony[**?**, **?**]. Ultimately Google unveiled its smartphone Nexus One that uses the Android open source mobile operating system. The device is manufactured by Taiwan's HTC Corporation, and became available on January 5, 2010.

On Feb 16, 2010 Google announced that 60,000 Android cell phones are shipping per day.

1.2 Hardware running Android

The first phone to run the Android operating system was the HTC Dream, released on 22 October 2008[**?**]. By the end of 2009 there will be at least 18 phone models using Android worldwide, according to Google[**?**]. In addition to

the mobile devices that ship with Android, some users have been able (with some amount of hacking, and with limited functionality) to install it on mobile devices shipped with other operating systems[**?**].

1.3 T-Mobile G1[?]

The HTC Dream (also marketed as T-Mobile G1 in the US and Europe [except for Spain, where it is marketed as HTC Dream] or Era G1 in Poland) is an Internet-enabled 3G smartphone with an operating system designed by Google and hardware designed by HTC. It was the first phone to the market that uses the Android mobile device platform. The phone is part of an open standards effort of the Open Handset Alliance.

processor The MSM7201A is an ARM-based, dual-core CPU/GPU from Qualcomm and contains many built-in features, including 3G and a GPU capable of up to 4 million triangles/sec.

memory The HTC Dream has a microSD card slot and comes with a 1GB memory card (2GB in the UK, Germany and Canada). It has been confirmed to work with capacities up to 16GB, and may work with even larger cards.

secondary storage N/A

RF sub-system screen 3.2 in (8.1 cm) TFT-LCD flat glass touch-sensitive HVGA screen with 480 X 320 pixel resolution.

camera The HTC Dream has a 3.2-megapixel camera with autofocus functionality sensors

GPS The HTC Dream provides an accelerometer for detecting movement and determining which direction is 'Up'. It also has a digital compass, giving it complete orientation data. The Dream includes a GPS receiver for fine-grained positioning, and can use cellular or wifi networks for coarse-grained positioning.

audio system The standard headset includes a clip-on microphone and call answer/hangup button. The Dream supports audio files in MP3, AAC, AAC+, WMA, MPEG4, WAV, MIDI, and Ogg formats.

battery The HTC Dream has a user-replaceable, 3.7V, 1150 mAh (4.25Whr) rechargeable lithium ion battery, which is advertised to offer up to 130 hours of standby power.

1.4 Thread management system

In Android, every time when user start up a application, it will generate a Linux process and a main thread. Normally, all components of an ongoing application are maintained in this process and thread. If resources are running out, Android would try to stop some threads to keep running this application. The most interesting things that I am care about the Android thread management is how to conduct a painless thread within a single thread model. Android is using a single thread model: when you first start an Android application, a thread called "main" is automatically created. The main thread, also called the UI thread, is very important because it is in charge of dispatching the events to the appropriate widgets and this includes the drawing events. It is also the thread

you interact with Android widgets on. For instance, if you touch the a button on screen, the UI thread dispatches the touch event to the widget which in turn sets its pressed state and posts an invalidate request to the event queue. The UI thread dequeues the request and notifies the widget to redraw itself.

However, this single thread model could get involved into poor performance, because the UI thread could be blocked by the ongoing tasks, such as network and etc. Thus, the sub-threads' tasks should not be handled by main UI thread. To solve this problem, Android introduces a method that combing Message queue, handler and looper to exchange messages between threads.

- *Message Queue* is involved to store the messages published by handler. It is maintained by the thread.
- Through *Handler*, an instance could be sent. Every handler is related with unique thread and its message queue. When a Handler is created, it is bound to the message queue of the thread that created it. On default, it will only associated with the threads that created. That means, if the threads send message to others, it will only be sent to the associated message queue.
- *Looper* is like a "bridge" between handler and message queue. Application threads first put the message to looper through handler, then looper put the message into message queue. Meanwhile, looper broadcast the message to all handlers for the threads.

Besides, Android introduces "AsynTask" when UI threads are needed to accessed and updated frequently.

1.5 Interrupts

When interrupt happens in Android, signals is sent that a blocking I/O operation has been interrupted. A field of the exception kept all the number of bytes that were transferred successfully before the interruption took place. Lock implementations provide more extensive locking operations than can be obtained using synchronized methods and statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated Condition objects. As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread.

For example, if a broadcast is to happen, a call is made to the activity manager (possibly going back down through native driver, kernel and back up in the system process). Then for a broadcast in the activity manager we find all of the interested receivers of the broadcast through the package manager and internal state in the activity manager, and dispatch those through further IPCs to the processes they live in.

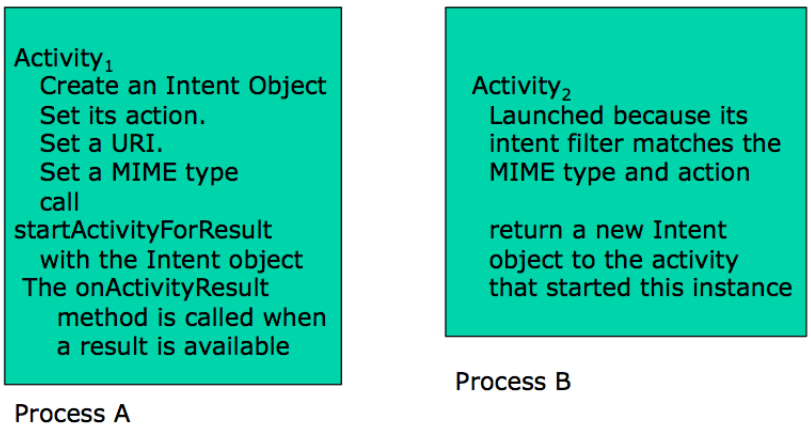

Figure 2: IPC-Intent

1.6 Interprocess communication

In computing, Inter-process communication (IPC) is a set of techniques for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network. IPC techniques are divided into methods for message passing, synchronization, shared memory, and remote procedure calls (RPC). The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

Generally speaking, interprocess communication approaches in Android could be divided into two categories: intent and remote methods. An intent is an abstract description of an operation to be performed. The intent approach is shown in Figure **??**.

The Remote method is using AIDL, shown in Figure **??**. AIDL (Android Interface Definition Language) is an IDL language used to generate code that enables two processes on an Android-powered device to talk using interprocess communication (IPC). If you have code in one process (for example, in an Activity) that needs to call methods on an object in another process (for example, a Service), you would use AIDL to generate code to marshall the parameters. The AIDL IPC mechanism is interface-based, similar to COM or Corba, but lighter weight. It uses a proxy class to pass values between the client and the implementation.

1.7 System calls

In computing, a system call is the mechanism by which a program requests a service from an operating system's kernel. In Android, system calls is dispatched from the applications' threads.

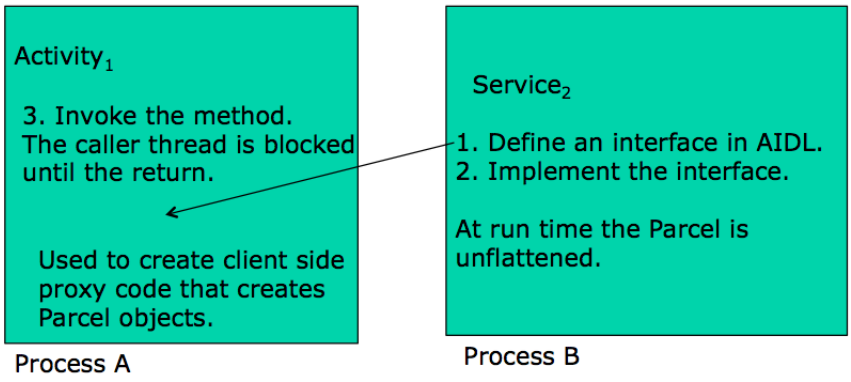

Figure 3: IPC Remote method

1.8 Memory management

Basically, Android manages its memory automatically by Dalvik garbage collector. It will recollect memory resources when ongoing application is running out the memory based on the level of importance. However, it introduces a serious problem. When many allocations happen, the garbage collector will kick in and stop the user's application to let it free some memory. Most of the time, garbage collections happen fast enough for you not to notice. However, if a collection happens while you are scrolling through a list of items or while you are trying to defeat a foe in a game, you may suddenly see a drop in performance/responsiveness of the application. It's not unusual for a garbage collection to take 100 to 200 ms. For comparison, a smooth animation needs to draw each frame in 16 to 33 ms. If the animation is suddenly interrupted for 10 frames, you can be certain that your users will notice.

To avoid this problem, the Android SDK ships with a very useful tool called allocation tracker, which is a part of DDMS, to track the application's memory and guarantee correctness of the program.

1.9 Networking support (Wi-Fi, Bluetooth, 3g, 4g)

Wi-Fi The Wi-Fi APIs provide a means by which applications can communicate with the lower-level wireless stack that provides Wi-Fi network access. Almost all information from the device supplicant is available, including the connected network's link speed, IP address, negotiation state, and more, plus information about other networks that are available. Some other API features include the ability to scan, add, save, terminate and initiate Wi-Fi connections.

Bluetooth The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth functionality through the Android Bluetooth APIs. These APIs let applications wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

1.10 Power management system

Android implements a very simple power management mechanism. Basically speaking, it utilizes locks and timer to maintain a relatively low level of power consumption. Currently it only supports set screen on/off, screen backlight on/off, keyboard backlight on/off, button backlight on/off and adjust screen brightness. It does not support Sleep or Standby mode to fully use CPU's capability. The architecture overview is shown in Figure 2.

The power management module has three channels to receive input: RPC call, Batter state change event and Power Setting change event. It communicated with other modules through either broadcasting intent or directly API call. The module also provide reboot and shutdown service. When battery is lower than thredshold, it will automatically shutdown the device.

The module will automatically set screen dim and off according to whether any user activity happens or not. The system containing three states, as shown in Figure 3:

1. USER_AWAKE, it means that the system is "Full on status".
2. USER_NOTIFICATION, shows that the monitor is suspended driver but CPU keep on
3. USER_SLEEP means that CPU enter sleep mode

1.11 Software development kits

Android provides software development kits for developers to implement various apps. The Android Software Development Kit includes the Android system files, packaged APIs, and Google APIs add-ons. SDK Tools is a downloadable component for the Android SDK. It includes the complete set of development and debugging tools for the Android SDK

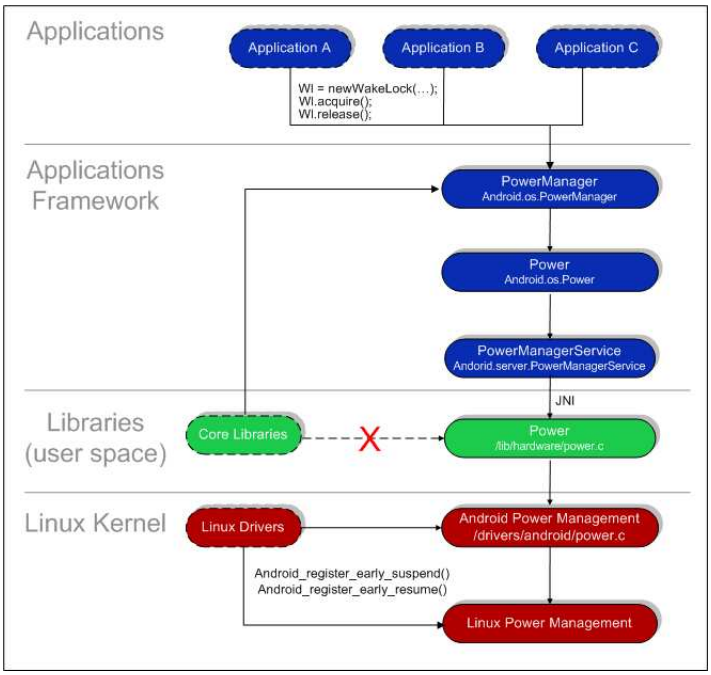

Figure 4: Android Power Management Overview

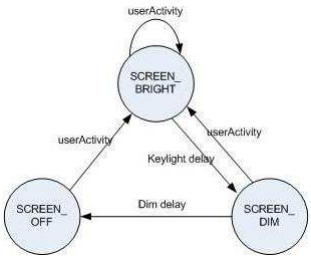

Figure 5: State Change

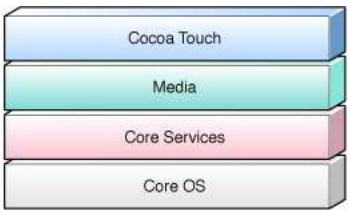

Figure 6: iPhone Layer Overview

2 iPhone OS

iPhone OS (known as OS X[?], or iPhone OS X in its early history) is a mobile operating system developed and marketed by Apple Inc. It is the default operating system of the iPhone, the iPod Touch, and the upcoming iPad. It is derived from Mac OS X, with which it shares the Darwin foundation, and is therefore a Unix-like operating system by nature. iPhone OS has four abstraction layers: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer. The operating system uses less than 500 megabytes of the device's memory.[?]

- The *Core OS and Core Services layers* contain the fundamental interfaces for iPhone OS, including those used for accessing files, low-level data types, Bonjour services, network sockets, and so on. These interfaces are mostly C-based and include technologies such as Core Foundation, CFNetwork, SQLite, and access to POSIX threads and UNIX sockets among others.
- The *Media layer* contains the fundamental technologies used to support 2D and 3D drawing, audio, and video. This layer includes the C-based technologies OpenGL ES, Quartz, and Core Audio. It also contains Core Animation, which is an advanced Objective-C based animation engine. It is uses a mixture of C-based and Objective-C based interfaces.
- The *Cocoa Touch layer* provide the fundamental infrastructure used by your application. For example, the Foundation framework provides object-oriented support for collections, file management, network operations, and more. It is based on Objective-C.

2.1 A brief review of the history of iPhone

The version history of iPhone OS began with the release of the iPhone on June 29, 2007. This operating system from Apple Inc. did not have an official name until the release of the iPhone SDK on March 6, 2008. Before then, Apple marketing literature simply stated that the iPhone runs "OS X", a reference to iPhone OS's parent, Mac OS X. The current version of iPhone OS is 3.1.3. The Developer Beta for version 3.0 was made available on March 17, 2009; iPhone OS 3.0 was released June 17, 2009

Apple provides updates to the iPhone's, iPad's and iPod Touch's operating system through iTunes, similar to the way that other iPods are updated, and touts this as an advantage compared to other mobile phones and devices.[3] Security patches, as well as new and improved features, are released in this fashion.

Apple concurrently provides the same version of the iPhone OS for the iPod Touch. iPhone users receive all software updates for free, but iPod Touch users are charged for major software updates like 2.0 and 3.0.

On October 17, 2007, in an open letter posted to Apple's "Hot News" weblog, Steve Jobs announced that a software development kit (SDK) would be made available to third-party developers in February 2008.[**?**] The SDK was released on March 6, 2008, and allows developers to make applications for the iPhone and iPod Touch, as well as test them in an "iPhone simulator". But developers have to pay to load applications onto the devices. Developers are able to set any price above an minimum for their apps in the App Store, of which they will receive a 70% share.

2.2 Hardware

processor Intel XScale-PXA27x rev 7 (v5l); Original & 3G: Samsung 32-bit RISC ARM 1176JZ(F)-S v1.0[6]; 620 MHz underclocked to 412 MHz[7]; PowerVR MBX Lite 3D GPU[8]; 3GS: Samsung S5PC100 ARM Cortex-A8[9]; 833 MHz underclocked to 600 MHz; PowerVR SGX GPU[10]

bus

memory Original & 3G: 128 MB eDRAM 3GS: 256 MB eDRAM secondary storage

RF sub-system

screen 320 × 480 px, 3.5 in (89 mm), 2:3 aspect ratio, 18-bit (262,144-color) LCD with 163 pixels per inch (ppi)

camera Original & 3G: 2.0 megapixels with geotagging 3GS: 3.0 megapixels with video (VGA at 30 fps), geotagging, and automatic focus, white balance, & exposure

sensors When the user lift iPhone to his ear, the proximity sensor immediately turns off the display to save power and prevent accidental dialing. The ambient light sensor in iPhone automatically brightens the display when you're in sunlight or a bright room and dims it in darker places.

GPS iPhone 3GS can find users' location quickly and accurately via GPS, Wi-Fi, and cellular towers. GPS (Global Positioning System) technology uses information from earth-orbiting satellites to find locations. A-GPS (Assisted GPS) on iPhone 3GS goes a step further, finding the closest satellites to more quickly identify user's position. There is a build-in digital compass that works just like a magnetic needle compass that can let users know which way they are facing.

audio system The audio processing plug-ins, known as audio units, can be dynamically accessed from the user's iPhone OS application. It allows user to add a variety of useful, prepackaged audio features and take advantage of the low

latency that audio units offer. iPhone OS ships with audio units that support mixing, equalization, format conversion, and I/O for recording, playback, or live chat.

battery Li-ion 1000 mAh

2.3 Thread management system

Each process (application) in Mac OS X or iPhone OS is made up of one or more threads, each of which represents a single path of execution through the application's code. Every application starts with a single thread, which runs the application's main function. Applications can spawn additional threads, each of which executes the code of a specific function.

When an application spawns a new thread, that thread becomes an independent entity inside of the application's process space. Each thread has its own execution stack and is scheduled for runtime separately by the kernel. A thread can communicate with other threads and other processes, perform I/O operations, and do anything else you might need it to do. Because they are inside the same process space, however, all threads in a single application share the same virtual memory space and have the same access rights as the process itself.

Threads itself in iPhone OS costs allocation memories from kernel and the programer's space. Creating threads is involving different methods, such as using NSThread, POSIX Threads and etc. Threads created using these techniques inherit a default set of attributes, determined by the technology you use. To terminate a thread, the recommended way is to let it exit its entry point routine normally, rather than killing threads explicitly because this would potentially result in memory leak.

2.4 Interrupts

Interrupt in iPhone OS are taken by ISR(Interrupt Service Routine), a function (subroutine) located within the code for the iPhone OS, acknowledges the interrupt and begins to process it via the corresponding driver. It is set up during driver initialization. For example, if firmware detects a hardware change, it will sends a processor interrupt through the system bus for attention. The iPhone's ARM processor receives the interrupt, and calls an in-memory ISR. The OS itself sends a signal to the current active application (if any) in the form of a Unix-style signal, which will be handled according to the specifications of the C/Objective-C runtimes. The Objective-C runtime forwards the signal as a framework-specific message to the application, first checking whether the current application has been designed to handle said message.

2.4.1 Interprocess communication

Originally, iPhone forbids the back-ground process and rejects apps that attempting to read databases and media folder directly based on the concern of

restricting and confining third-party apps to their sandbox. So IPC is basically not allowed. However, Apple approved to support "near"-inter-process communication using *URL protocol handler*.

First, the programmer needs to register this handler((like myapp://....) with iPhone. Then, when other apps calls it, the app would be launched and handle this call.

2.4.2 System calls

System calls in iPhone is achieved by the combination of API, framework and library calls. It is for providing a high-level, developer-friendly interface to UNIX kernel(XNU).

To begin with, the frameworks/API is the set of functions the developer wants to see in order to make his/her application interact with the device and underlying iPhone OS. The framework, which is part of the overall Objective-C runtime, is a series of upper-level function calls (and C extensions, in the case of the Obj-C runtime) in order to allow a more developer-friendly interface to the grimy, obfuscated (and undocumented) C library.

Then, the C library lies underneath the Objective-C runtime. It provids true object oriented dynamic-typing extensions to the C language (unlike C++, which just looks object-oriented). The C library is dynamically linked to from the Objective-C runtime to ÒtranslateÓ messages requiring lower-level intervention into UNIX system calls, where the iPhone OS can process them as needed.

For example, when we use graphics display. First, the application is loaded into memory and starting its execution, wishes to display an image for a splash screen. Secondly, it makes API (framework) call. The API/framework receives the function call, and does the dirty work of translating the higher-level image processing interface into a collection of calls to the Objective-C runtime, which will then make the appropriate calls to the C library. The (dynamically-linked) C library, having received a series of function calls from the Objective-C runtime and corresponding frameworks and APIs, will further refine the functions into assembly-level system calls (via a software interrupt) which the iPhone OS kernel can process. The iPhone OS kernel, having received the system calls in assembly format from the C library, will then call upon the appropriate drivers (the touchscreen display drivers, in this case) to interact with the hardware.

2.5 Memory management

Memory management is the programming discipline of managing the life cycles of objects and freeing them when they are no longer needed. In iPhone OS, the garbage collection is not available. Therefore, a mechanism should be introduced that allows you to mark an object as still being useful. In many respects, memory management is thus best understood in terms of "object ownership", which includes,

1. An object may have one or more owners.

2. When an object has no owners, it is destroyed.

3. One mush become the owner to make sure if an object interested is destroyed.

4. To allow an object in which youÕre no longer interested to be destroyed, you relinquish ownership.

To support the model, the Cocoa provides a mechanism called "reference counting". Every object has a retain count. An object is created with a retain count of 1. When the retain count drops to 0, an object is deallocated (destroyed). You manipulate the retain count (take and relinquish ownership) using a variety of methods, such as *alloc*, *copy*, *retain*, *release* and *autorelease*.

When one create or copy an object, its retain count is 1. Thereafter other objects may express an ownership interest in your object, which increments its retain count. The owners of an object may also relinquish their ownership interest in it, which decrements the retain count. When the retain count becomes zero, the object is deallocated (destroyed).

2.6 Networking support (Wi-Fi, Bluetooth, 3g, 4g)

The networking stack in iPhone OS includes several interfaces over the radio hardware of iPhone and iPod touch devices. The main programming interface is the CFNetwork framework, which builds on top of BSD sockets and opaque types in the Core Foundation framework to communicate with network entities.

Wi-Fi Wi-Fi, the IEEE 802.11 standard, is a two-way, short range protocol and operates in two bands. Initially, all Wi-Fi used 2.412-2.472 GHz. The 802.11n protocol added the ability to use 5.15 to 5.25 GHz (indoor use), but the iPhone is restricted to 802.11b/g.

Bluetooth Bluetooth is a two-way, ultrashort range protocol that works in the 2.40-2.485 GHz band. It avoids interference with other systems in the same band by being very low power, about a milliwatt, and has a maximum range of about 10 meters.

3G 3G technology gives iPhone fast access to the Internet and email over cellular networks around the world. With support for 7.2Mbps HSDPA, iPhone 3GS also makes it possible to do more in more places: Surf the web, download email, get directions, and watch video — even while user is on a call. Since iPhone seamlessly switches between EDGE, faster 3G, and even faster Wi-Fi, user can always get the fastest connection available. Users can even share theri Internet connection with laptop via Internet tethering.

4G 4G refers to the fourth generation of cellular wireless standards. It is a successor to 3G and 2G standards, with the aim to provide a wide range of data rates up to ultra-broadband (gigabit-speed) Internet access to mobile as well as stationary users. Verizon Wireless said earlier this year that they see the Apple iPhone on their 4G network in 2010. A rumor about Verizon and Apple already testing a new iPhone on the Verizon Wireless 4G LTE network sneaked its way through the Motorola Droid coverage over the weekend.

2.7 Power management system

iPhone OS use the same architecture of Mac OS X, yet it is not identical to the Mac. For example, iPhone do not have the power management toolkit as Mac OS does. Instead, this function is embeded into the core layer, which intelligently powers up planes of devices as the system goes into standby or to sleep.

2.8 Software development kits

With this software the programmer can develop applications that run on iPhone and iPod touch. Includes the Xcode IDE, Instruments, iPhone simulator, frameworks and samples, compilers, Shark analysis tool, and more. With over 1,000 new APIs, iPhone SDK provides developers with a range of new possibilities to enhance the functionality of their applications. New APIs also provide support for applications to communicate with hardware accessories attached to iPhone or iPod touch.

3 COMPARISON

3.1 development environments[?]

1) Language

- *Android*: Java
- *iPhone*: Objective-C

2) Programming Model

- *Android*: With Android's support for multiple processes and component reuse, the platform itself provides support for Intents and Activities (an Intent is just a variant of a command); provide a way of declaring user preferences in XML; XML format is extensible allowing custom UI components to be integrated
- *iPhone*: MVC design pattern, provide a way of declaring user preferences in XML; iPhone developers that wish to customize preferences will have to implement a UI from scratch

3) IDE

- *Android*: Android development leverages the excellent JDT tools; Everything Java is indexed, the IDE has a rich model of the source code, and refactoring is seamless; JDT's incremental compiler provides immediate feedback with errors and warnings as you type.

- *iPhone*: Xcode IDE, Instruments, iPhone simulator, frameworks and samples, compilers, Shark analysis tool, and etc.

4) UI Builder

- *Android*: Android UI builder can't display UIs how they'll actually appear.
- iPhone: iPhone app developers are given a good UI builder; It's flexible and can model some sophisticated UIs,

3.2 Ease to port third party applications

Android

Basically speaking, Android shares more in common with other Java platforms than with desktops with Desktop Linux. Rather than running desktop Linux PC software (which is built using the X11 "X Window System" paired with a window manager like KDE or GNOME) like Nokia's N900 running Maemo Linux, Android supplies a modified Java Virtual Machine similar in many respects to the BlackBerry OS and Symbian phones designed to run Java ME apps. Google has modified Android's Java bytecode interpreter (which it calls Dalvik) to avoid having to pay Sun licensing fees for the official JVM in Android. This enables Google to offer Android for free, and without any interference from Sun. It also effectively makes Android a Java platform, not a Linux platform.

One fundamental characteristic of Android that is both a strength and a weakness is its insular nature. Android's unique userspace stack offers no compatibility glide path for porting applications to and from conventional Linux environments, but it does offer a significantly higher degree of cohesion across devices, which means less fragmentation and a more predictable target for third-party software developers.

Many views Android as a compelling middle ground between conventional mobile Linux stacks—where permissiveness and modularity have resulted in significant fragmentation—and proprietary vertical stacks like Windows Mobile, which are more rigid and lack mutability. Android provides a highly structured and consistent solution, but it also provides enough flexibility to make it easy for carriers and device makers to differentiate their products and customize the user experience.

iPhone

Apple has taken an entirely different approach to delivering its mobile software platform. Rather than building a bytecode interpreter based upon a specific, customized implementation of Java ME, Apple introduced the iPhone running a scaled down version of its desktop Mac OS X Cocoa development environment. This leverages the installed brain trust of the company's Mac developers rather than the installed base of Java ME coders in the existing smartphone market.

It's still possible to port Java code to the iPhone, but it requires more translation work as Apple only supports Objective-C/C as an iPhone development language in its own tools. Rather than allowing iPhone developers to easily port over desktop Mac apps to the iPhone, the great overlap between iPhone and Mac development tools appears to have been more of strategy to draw developer attention to the Mac. Apple already sells about twice as many iPhones as it does Macs, and the iPhone certainly casts a larger mindshare net than the Mac platform does itself.

Comparison

The Android or iPhone software platform is more than just a core operating system. And really, the differences in their core operating systems are one of the least important factors to users. Both use a Unix-derived kernel and operating system environment that few users will ever even see. Android phones happen to use a Linux kernel while the iPhone uses the same Mach/BSD Unix kernel as Apple's desktop Macs. The differences between developing for Android and for the iPhone are not a clear win for either camp. Both offer somewhat similar tools in terms of capability, with Google's being more familiar to open source or Java developers and Apple's being nearly identical to its desktop Mac platform tools. Apple has a minor lead in having deployed its platform about a year and a half before Android reached the market, and because it has been actively working on its Mac OS X platform for over a decade; Google is new to the platform development business. In addition to their internal technical differences, Android and the iPhone platform also differ in many other more significant respects that will more directly impact users. It's possible to use inferior technology to create a good product, and to use excellent technology to deliver a terrible product. More than technical specifics, users will be most impacted by platform factors such as:

- User restrictions and/or freedoms accorded by the platform's business model.
- The potential for rapid advancement of new features, increased sophistication and greater performance delivered in software updates.
- The usability of core bundled applications and the availability and affordability of useful and desirable third party software.

3.3 A discussion of virtualization

Virtualization is a broad term that refers to the abstraction of computer resources so that the development cost could be reduced in a rather considerable count. Mobile virtualization brings to devices is a thin layer of software that is embedded on the phone to decouple the operating system, applications and data from the underlying hardware. Despite obvious similarities between enterprise/desktop virtualization and its mobile counterpart, mobile phone use cases

present key differences: smaller memory capacities demand slimmer embedded hypervisor footprints, current mobile processors lack virtualization support in hardware requiring paravirtualization, and hosted guest software span the gamut from enterprise OSes to embedded RTOSes to stand-alone device drivers.

Since Android is an open-source system, virtualization could be done on it and it varies from different companies or organizations, such as OK Lab, VMware and etc. As claimed by VMware, they have done virtualization of running Android and Windows CE in a single mobile phone. However, iPhone does not have this virtualization based on its close-source situation.

3.4 Reliability and security

Android

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.

As an open platform, Android allows users to load software from any developer onto a device. As with a home PC, the user must be aware of who is providing the software they are downloading and must decide whether they want to grant the application the capabilities it requests. This decision can be informed by the user's judgment of the software developer's trustworthiness, and where the software came from.

iPhone

iPhone has no security software and Apple doesn't let people load third-party programs on the device, which could reduce the risk of infection from malicious software. When the iPhone is connected to the Web, dangerous possibilities emerge.

The iPhone Auto-Lock disables the device's screen after a preset time period of non-use, but the Passcode Lock feature takes that a step further. Whenever the device's display locks, whether due to Auto-Lock or because you've hit the iPhone Sleep button–found on the top right of the device–Passcode Lock requires a four-digit code to be entered before the device can be employed again.

The iPhone OS security APIs are located in the Core Services layer of the operating system and are based on services in the Core OS (kernel) layer of the operating system. Applications on the iPhone call the security services APIs directly rather than going through the Cocoa Touch or Media layers. Networking applications can also access secure networking functions through the CFNetwork API, which is also located in the Core Services layer.

The iPhone OS security implementation includes a daemon called the Security Server that implements several security protocols, such as access to keychain items and root certificate trust management.

The Security Server has no public API. Instead, applications use the Keychain Services API and the Certificate, Key, and Trust services API, which in turn communicate with the Security Server. Because, unlike the Mac OS X security services, the iPhone OS security services do not provide an authentication interface, there is no need for the Security Server to have a user interface.

Although Mac OS X includes a low-level command-line interface to the OpenSSL open-source cryptography toolkit, this interface is not available on the iPhone OS. For iPhone OS development, use the CFNetwork API for secure networking and the Certificate, Key, and Trust Services API for cryptographic services.

3.5 My consideration

The advantages and disadvantages of both systems is concluded in the following Table.

	Android	iPhone
Adv.	1. open-source, ease in third-party apps	1. Sufficient documentation
	2. multi-tasking	2. sophsiticated developemnt
	3. flexible	3. uniformed product
	4. can solve security issues	4. support multi-task after V4.0
	5. Can be virtualized	
Disadv.	1. versitile products	1. too many restrictions, not flexible
	2. insufficient documentation	2. not ease to third-party apps
		3. security issues
		4. can not be virtulized

References

[1] Elgin, Ben (2005-08-17). "Google Buys Android for Its Mobile Arsenal". Business Week. http: //www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm. Retrieved 2007-11-07.

[2] Block, Ryan (2007-08-28). "Google is working on a mobile OS, and it's due out shortly". Engadget. http://www.engadget.com/2007/08/28/google-is-working-on-a-mobile-os-and-its-due-out-shortly/. Retrieved 2007-11-06.

[3] Sharma, Amol; Delaney, Kevin J. (2007-08-02). "Google Pushes Tailored Phones To Win Lucrative Ad Market". The Wall Street Journal. http://online.wsj.com/article_email/SB118602176520985718-lMyQjAxMDE3ODA2MjAwMjIxWj.html. Retrieved 2007-11-06.

[4] "Google admits to mobile phone plan". directtraffic.org. Google News. 2007-03-20. http://www.directtraffic.org/OnlineNews/ Google_ admits_to_ mobile_phone_plan_18094880.html. Retrieved 2007-11-06.

[5] McKay, Martha (21 December 2006). "Can iPhone become your phone?; Linksys introduces versatile line for cordless service". The Record: p. L9. "And don't hold your breath, but the same cell phone-obsessed tech watchers say it won't be long before Google jumps headfirst into the phone biz. Phone, anyone?"

[6] Ackerman, Elise (2007-08-30). "Blogosphere Aflutter With Linux-Based phone Rumors". Linux Insider. http://www.linuxinsider.com/rsstory/59115.html. Retrieved 2007-11-07. [dead link]

[7] Claburn, Thomas (2007-09-19). "Google's Secret Patent Portfolio Predicts gPhone". InformationWeek. http://www.informationweek.com/news/showArticle.jhtml?articleID=201807587&cid=nl_IWK_daily. Retrieved 2007-11-06.

[8] Pearce, James Quintana (2007-09-20). "Google's Strong Mobile-Related Patent Portfolio". mocoNews.net. http://www.moconews.net/entry/ 419-googles-strong-mobile-related-patent-portfolio/. Retrieved 2007-11-07.

[9] T-Mobile Unveils the T-Mobile G1 - the First Phone Powered by Android". HTC. http://www.htc.com/www/press.aspx?id=66338&lang=1033. Retrieved 2009-05-19.

[10] Richtel, Matt (27 May 2009). "Google: Expect 18 Android Phones by Year's End". The New York Times. http://bits.blogs.nytimes.com/2009/05/27/google-expect-18-android-phones-by-years-end/. Retrieved 19 June 2009.

[11] "Android". Openmoko wiki. http://wiki.openmoko.org/wiki/Android. Retrieved 29 December 2009.

[12] http://en.wikipedia.org/wiki/HTC_Dream

[13] http://java.dzone.com/articles/android-vs-iphone-development

[14] Open Handset Alliance (2007-11-05). "Industry Leaders Announce Open Platform for Mobile Devices". Press release. http://www.openhandsetalliance.com/press_110507.html. Retrieved 2007-11-05.

[15] Kumparak, Greg (16 February 2010). "Google: Android now shipping on 60,000 handsets per day". MobileCrunch. http://www.mobilecrunch.com/2010/02/16/google-now-shipping-60000-android-handsets-per-day/.

[16] Chris Ziegler (January 9, 2007). "The Apple iPhone". Engadget. http://www.engadget.com/2007/01/09/the-apple-iphone/. Retrieved March 14, 2010.

[17] Haslam, Karen (January 12, 2007). "Macworld Expo: Optimised OS X sits on 'versatile' flash". Macworld. http://www.macworld.co.uk/ipod-itunes/news/index.cfm?newsid=16927. Retrieved 2007-10-15.

[18] Jobs, Steve (2007-10-17). "Third Party Applications on the iPhone". Apple Inc.. http://developer.apple.com/iphone/devcenter/third_party_apps.php.