Comparison of Android and iPhone Operating System

Alaa Nazeeh Mohmedhussen
Master in Computer Applied Technology
Diyala University, Iraq

Wahhab Isam Altaee
Master in Computer Applied Technology
Diyala / Iraq

ABSTRACT
This technical paper contains the information about the most commonly used smart phones and the operating systems that are used by these smart phones. After this a brief introduction of the android and IOS operating system has been described. A basic comparison between these has been started by. The technical comparison has been made by describing the core architecture of applications and the system management, inter-process communication, system calls, virtualization, memory management and power management. The comparisons of these Operating Systems have been made by different aspects that clarify the main differences between both.

Keywords
Android, IOS, Architecture, Operating System, comparison

1. INTRODUCTION

1.1 Smartphone Operating Systems
There are many Operating Systems for smart phones. The main mobile operating systems (OS) used by modern smart phones includes the following:

- Google's Android
- Apple's iOS
- Nokia's Symbian,
- RIM's BlackBerry OS
- Samsung's Bada
- Microsoft's Windows Phone
- Hewlett-Packard's web OS,

Such operating systems can be installed on many different phone models, and typically each device can receive multiple OS software updates over its lifetime.

1.2 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.

2. SYSTEM ARCHITECTURE

2.1 Android Architecture
The Android architecture has several layers as showed in Figure 1.

Kernel: Linux is support 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.

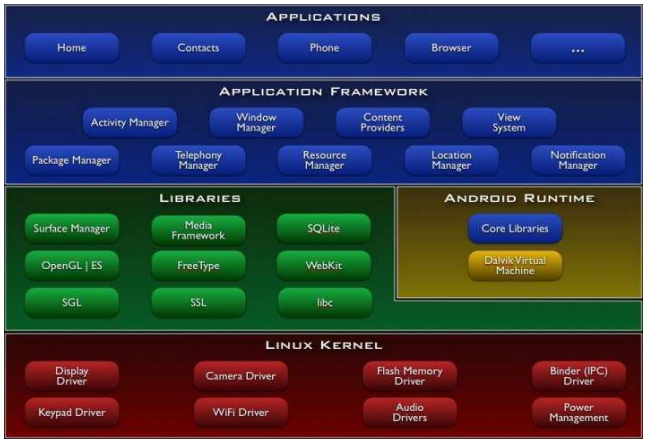

Fig 1: Android Architecture

Runtime: Runtime includes core libraries and Dalvik virtual machine. Core libraries have 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. 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.

Libraries: Android has a set of C/C++ libraries used by various components of the Android system. These libraries are exposed to developers. The system C library is a BSD-derived implementation of the standard.

System library. The media libraries are based on Packet Video's open CORE which supports playback and recording of many popular audio formats. The surface manager manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.

Application Framework: All Android applications are written with Java programming language, it ships with a set of core applications including email client, SMS program, calendar, maps, browser, contacts and Others. Android offers developers the ability to build various applications with an open development. 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.

This same mechanism allows components to be replaced by the user. The applications including views that can be used to build an applications, including lists, grids, text boxes, buttons, and even an embeddable web browser. Content providers enable applications to access data from other applications or to share data with others. The resource manager providing access to non-code resources such as localized strings, graphics and layout.

2.2 IPhone Architecture

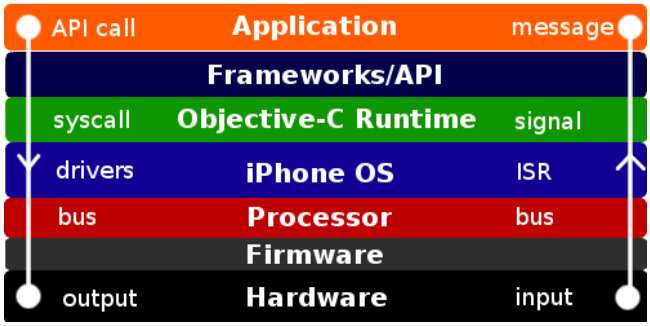

Fig. 2: iPhone Architecture

Hardware: In iPhone, Hardware refers to the physical chips soldered to the iPhone's circuitry. The actual processor falls under this layer, but the instruction set and in- memory descriptor tables are contained within the "processor" layer.

Firmware: Firmware refers the chip-specific code that is either contained with memory in/around the peripheral itself, or within the drive for said peripheral. Processor:

Processor is refers to the ARM instruction set and the interrupt descriptor table as set up by the iPhone OS during boot and driver initialization.

iPhone OS: iPhone OS is the kernel, drivers, and services that comprise of the iPhone

Operating System. It sits between the user space and hardware.

Objective-C Runtimes: Objective-C runtime is comprised of both the Objective-C dynamically-linked runtime libraries, as well as the underlying C libraries. Frameworks/API: **Frameworks/API layer** has API calls which are Apple-distributed headers with the iPhone SDK, with some dynamic linking occurring at runtime. These reside on top of the Objective-C runtime, as many of these are written in Objective-C.

Application: The application stored in iPhone has to be purchased through the application store. The Application was compiled to native code by the Apple- distributed iPhone compiler, and linked with the Objective-C runtime and C library by the linker. The application also runs entirely within the user space environment set up by the iPhone OS.

3. SYSTEM MANAGEMENT

3.1 Android

When the first of an application's components needs to be run, Android starts UI thread, a Linux processor it with a single thread of execution. The UI thread, which also know as main thread is automatically Created when an Android application is started. UI thread is very important because it takes in charge of dispatching the event to the appropriate widgets. UI thread also the thread people interact with the Android widgets. When people touches a button, the UI thread will dispatches the touch event to the widget which in turn sets its pressed state and posts an invalidate request to the event queue. It dequeues the request and notifies the widget to redraw itself. By default, all components of the application run in that process and thread. Each component is run by a process, and the component elements have a process attribute which specify the process where the component should run. By setting bthe attributes in different way, it is easy to know whether the component run in its own process or share a process with other components. No component should perform long operation when called by the system in order not to block other components which also in the process. Besides, when the memory is low and other processes have more immediately requirement, the Android will decide to shut down a process. The application components running in that process will be destroyed consequently. Therefore, android weights the importance of each thread to the user to decide the victim.

Android provides Handler and Looper to manage threads and let them communicate with each other.

Looper is used for running a message loop within a thread, **headler** is processing the messages and

HandlerThread is setting up a thread with a message loop. After a child thread create an image from the web, it notifies the UI thread by sending a message using the handler that's bound to UI thread's message queue. The data produced by the child process can also send via the message. Therefore, the UI thread can update with the data produced by the child thread. When a Handler is created, it is bound to the message queue of the thread that created it.

3.2 IPhone

In iPhone OS, each application is made up of one or more threads, which represents a single path of Execution . Every applications starts with a single thread, which runs the application's main function. The applications can have additional threads with executes a specific function. When application generates a new thread, it becomes an independent entity inside the process space. Each thread has its own execution stack and the kernel schedules its run time separately. As threads are in the same process space they can communicate with other threads and processes. All threads in a single application share the same virtual memory space and have the same access rights as the process itself. Each thread requires the memory allocation in both the kernel memory space and the program memory space. The core structures needed to manage the thread and coordinate its scheduling. The stack space of thread and the per-thread data is stored in program's memory space. When first create the thread, the most of structures are created and initialized. In order to creating the low-level thread, a function or method to act as the main entry point for thread is needed, then one of the available thread routines has to be used to start the thread. The thread can be created by using the different methods as using NSThread, POSIX Threads, or using NSObjec to spawn a thread. After a thread is created, different thread environment needed to be configured, like configuring the stack size of a thread and the thread-local storage, setting the detached state of a thread and the thread priority.

4. INTER-PROCESS COMMUNICATION

4.1 Android

In Android's computing, the 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. 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. tabs, and so on. For three authors, you may have to improvise.

4.2 IPhone

For iPhone, any time the sequence in which two operations are completed affects the result, there is the potential for a race condition. For example, if two processes (in a single program or different programs) share the same global variable, then there is the potential for one process to interfere with the other or for an attacker to alter the variable after one process sets it but before the other reads it. The solution to race conditions of this type is to use some locking mechanism to prevent one process from changing a variable until another is finished with it. There are problems and hazards associated with such mechanisms, and they must be implemented carefully.

5. SYSTEM CALLS

A system call is a request made by any program to the operating system for performing tasks—picked from a predefined set—which the said program does not have required permissions to execute in its ownflow of execution. System calls provide the interface between a process and the operating system.

5.1 Android

Developer can trace the system calls invoked by Android for specific tasks by Android Task. The developer have to figure out which process is actually managing the data items, and then use strace <app name>after start to strace an application. It can also attach to an existing process by using the command 'strace -p <pid>'. Use 'ps' to find the process which is managing the contacts and messaging.

5.2 iPhone

The iPhone OS provides tools for system profiling. The Shark application lets user selectively find out what file system-related function calls the application makes. When user set up the session configuration, he can tell Shark exactly which function calls you want it to watch. A list of system calls includes all the file I/O calls plus fcntl, flock, fstat, fsync, link, lstat, lstatv, and stat. The sc_usage tool displays an ongoing sample of system statistics for a given process, including the number of system calls and page faults. The tool adds new system calls to the list as they are generated by the application being watched. The counts displayed are both the cumulative totals since sc_usage was launched and the delta changes for this sample period.

6. MEMORY MANAGEMENT

Memory management is the programming discipline of managing the life cycles of objects and freeing. Them when they are no longer needed. Managing object memory is a matter of performance; if an application doesn't free unneeded objects, its memory footprint grows and performance suffers.

6.1 Android

Android applications usually are limited to 16 MB of heap. Though it is very little for some developers, it isa lot of memory for a phone. Because the more applications Android can keep in memory, the faster it when user switch between its applications. Therefore, the applications should use as little memory as possible to guarantee multiple applications run without getting killed. The memory leak phenomenon affects the memory usage, and hence affects the application switch efficiency. All Android apps are written in Java. Java, unlike other programming languages, does not require any coding to allocate or deal locate memory and handles all memory allocation/DE allocation through a "feature" called automatic garbage collection. The garbage collector kicks in whenever an application no longer has any active threads pointing to it. By design, the garbage collector will not remove applications that are actively being used; but there may be a bit of system degradation while things are being cleaned up (usually a very fast process). Android rely on automatic memory management which handled by garbage collector. However, the garbage collector can sometimes cause performance issues if memory allocation is not handled carefully. The Android SDK provides allocation tracker, a tool to avoid the frequent garbage collection.

6.2 IPhone

IPhone has no garbage collection, developer has to clean up the variables after use them, otherwise the program will leak memory. Though NSObject class has accounting stuff help to keeps the track of how many other objects are currently using the object, but it is not automatic and developers have to adjust that by themselves. The rule for managing memory is to make sure the number of ownership methods called on an object will equal the number of loss-of-ownership methods by the time the program has finished executing. When create or copy an object, its retain count is

1. Thereafter other objects may express an ownership interest in your object, which increments it's 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 deal located (destroyed).

7. POWER MANAGEMENT

7.1 Android

Android supports its own Power Management (on top of the standard Linux Power Management) designed with the premise that the CPU shouldn't consume power if no applications or services require power. Android requires that applications and services request CPU resources with "wake locks" through the Android application framework and native Linux libraries. If there are no active wake locks, Android will shut down the CPU. The Android Framework exposes power management to services and applications through the Power Manager class. User space native libraries should never call into Android Power Management directly. Bypassing the power management policy in the Android runtime will destabilize the system. All calls into Power Management should go through the Android runtime Power Manager APIs.

7.2 iPhone

IPhone do not have the power management toolkit as Mac OS does. Instead, this function is embedded into the core layer, which intelligently powers up planes of devices as the system goes into standby or to sleep. The most power hungry systems in iPhone from most to least are 3G radio system, Wi-Fi, 2G radio system, Bluetooth and GPS. Reduce the number of power consuming applications helps to save the energy. When put an iPhone into sleep, it will disconnect from network, turn off the Wi-Fi and screen light.

8. COMPARISON

8.1 Development Environments

The Android use java as develop language. 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 command) and provides a way of declaring user preferences in XML which is extensible allowing customer components to be integrated. 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. In Android, UI builder can't display UIs how they'll actually appear. iPhone uses Objective-C as its development language. Eclipse can be used as iPhone OS development IDE.

Beside, Aptana, a cross-platform IDE has released their iPhone Development Plug-in for their Ajax IDE. It can preview pages in both horizontal and vertical mode, and includes a wizard for importing the most common Ajax libraries. iPhone application developers are given a good UI builder; It's flexible and can model some sophisticated UIs, ease to port third party applications

1) Language

- ✓ Android: Java
- ✓ IPhone: Objective-C

2) Integrated Development Environment IDE

A) **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.

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

3) Programming Model

A) **Android:** With Android's support for multiple processes and component reuse, the platform itself provides support for Intents and Activities (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.

B) **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.

4) UI "User Interface" Builder

A) **Android:** Android UI builder can't display UIs how they'll actually appear.

B) **IPhone:** iPhone app. developers are given a good UI builder; It's flexible and can model some sophisticated UIs.

8.2 Reliability and Security

8.2.1 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.

8.2.2 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 CF Network API, which is also located in the Core Services layer.

8.3 Virtualization

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 .dexformat by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management. Unlike most virtual machines and true Jave VM switch are stack machine, the Dalvik VM is a register-based architecture. Like the CISC vs. RISC debate, the relative merits of stack machine vs. register-based approaches are a subject of continual argument. Generally, stack-based machines must use instructions to load data on the stack and manipulate that

data, thus, require more instructions than register machines to implement the same high level code, but the instructions in a register machine must encode the source and destination registers and, therefore, tend to be larger. This difference is primarily of importance to VM interpreters for whom opcode dispatch tends to be expensive and other factors are relevant for JIT compilation.

A tool called dx is used to convert Java .clasee files into the .dex format. Multiple classes are included in asingle .dex file. Duplicate strings and other constants used in multiple class files are included only once in the .dex output to conserve space. Java bytecode is also converted into an alternate instruction set usedby the Dalvik VM. An uncompressed .dex file is typically a few percent smaller in size than compressed .jar (Java Archive) derived from the same .class files.

The Dalvik executables may be modified again when they get installed onto a mobile device. In order to gain further optimization, byte order may be swapped in certain data, simple data structures and function libraries may be linked inline, and empty class objects may be short-circuited, for example.

Being optimized for low memory requirements, Dalvik has some specific characteristics that differentiate it from other standard VMs: The VM was slimmed down to use less space. Dalvik currently has no just-intimecompiler, but Android 2.0 includes experimental source for one (disabled by default). The constant pool has been modified to use only 32-bit indexes to simplify the interpreter. It uses its own byte code, not Java byte code. Moreover, Dalvik has been designed so that a device can run multiple instances of the VM efficiently. For iPhone, Sun Microsystems plans to release a Java Virtual Machine (JVM) for iPhone OS, based on the Java Platform, Micro Edition version of Java. This would enable Java applications to run on iPhone and iPod Touch.

This figure describes the market shares of operating systems of smart phones.

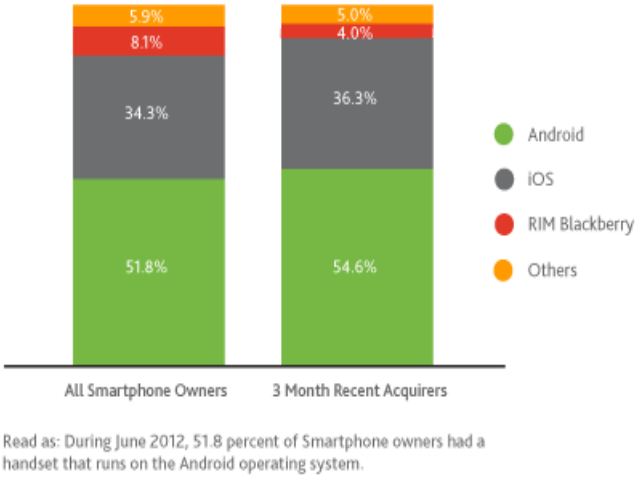

Fig 3: Market share of operating system

And this figure describes the difference between android and ios with respect to their usage**.**

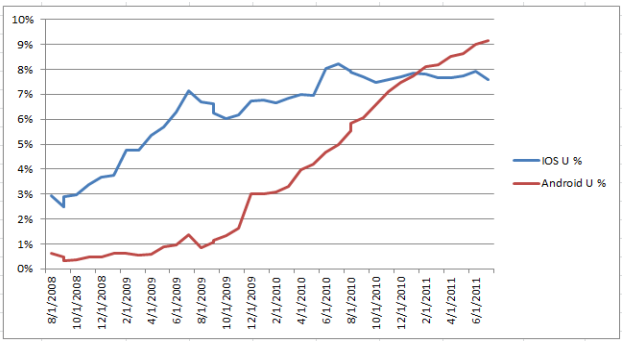

Fig 4: Different between android and ios

9. CONCLUSION

Conclusion of both competitors is defined as follow in form of advantages and disadvantages.

Table 1: competitors form of advantages and disadvantages

	Android	IOS
Advantages	1. Open-source, ease in third-party	1. Sufficient documentation
	2. Multi-tasking	2. Sophisticated development
	3. Flexible	3. Uniformed product
	4. Can solve security issues	4. Support multi-task after V4.0
	5. Can be virtualized	5.
Disadvantages	1. Versatile products	1. Too many restrictions, not flexible
	2. Insufficient documentation	2. Not ease to third-party apps
	3. If Apps are too long then it force	3. Security issues
	4. Can also slow down if installing	4. Cannot be virtualized

The Compare with iPhone, Android provides developer more freedom. As Android is an open source, developer can create and deploy anything on the device. Compare with iPhone OS, Android is more widespread and applied in different model. Besides, Google provides many services like Google Search, Talk, Google Doc, Google Map and so on. Once integrate these services with Android, it can provide customer more benefits .iPhone OS, on the contrary, is designed exclusively for Apple's products, like iPhone, iPod and iPad. This helps to avoid reduce the functionality to suit the hardware needs for different devices. Also this focus an help developers make more useful and powerful applications based on customers requirement.

10. REFERENCES

[1] Greg Kroah-Hartman. "Android and the Linux kernel community" http://www.kroah.com/log/linux/android-kernel-problems.html.

[2] "Linux developer explains Android kernel code removal". http://news.zdnet.com/2100-9595_22-389733.html Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[3] "What is Android". Android Developers. http://developer.android.com/guide/basics/what-isandroid.

[4] Open Handset Alliance. "Industry Leaders Announce Open Platform for Mobile Devices". http://www.openhandsetalliance.com/press_110507.html.

[5] Bort, Dave. "Android is now available as open source". Android Open Source Project. http://source.android.com/posts/opensource.

[6] Honan, Matthew. "Apple unveils iPhone". Macworld. http://www.macworld.com/article/54769/2007/01/iphone.htm

[7] Android architecture. http://www.slideshare.net/deepakshare/android-arch-presentation

[8] 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-isworking-on-a-mobile-os-and-its-due-out-shortly/. Retrieved 2007-11-06.

[9] Sharma, Amol; Delaney, Kevin J. (2007-08-02). "Google Pushes Tailored Phones To Win Lucrative Ad Market". The Wall Street Journal

[10] "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.

[11] Android vs iPhone. By Junyao Zhang. Available online http://www.cs.ucf.edu/~dcm/Teaching/COP5611Spring2010/Project/JunyaoZhang-Project.pdf