**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**

*IJCSMC, Vol. 4, Issue. 3, March 2015, pg.647 – 651*

**RESEARCH ARTICLE**

# DELTA++: Reducing the Size of Android Application Updates

**S.Natarajan**
Student
Dept of Computer Sccience Engg
IFET College of Engineering, Villupuram

**Mr.M.Pajany, M.E**
Associate Professor
Dept of Computer Science Engg
IFET College of Engineering, Villupuram

**ABSTRACT**

This method of creating and deploying update patches improves on Google Smart Application Update by first unpacking the Android Application Package and then compressing its elements individually. The smartphone user can then download a smaller patch. Experiments show that performance yields 49 percent more reduction relative to Google's solution, increasing the savings in cellular network bandwidth use and resulting in lighter application server loads. This reduction in Android application-update traffic could translate to a 1.7 percent decrease in annual US cellular traffic. Similar methods applied to iPhone application updates could yield even greater savings.

**INTRODUCTION**

Mobile computing is the discipline for creating an information management platform, which is free from spatial and temporal constraints. The freedom from these constraints allows its users to access and process desired information from anywhere in the space. The state of the user, static or mobile, does not affect the information management capability of the mobile platform. A user can continue to access and manipulate desired data while traveling on

plane, in car, on ship, etc. Thus, the discipline creates an illusion that the desired data and sufficient processing power are available on the spot, where as in reality they may be located far away. Otherwise **Mobile computing** is a generic term used to refer to a variety of devices that allow people to access data and information from where ever they are.

**PROPOSED SYSTEM**

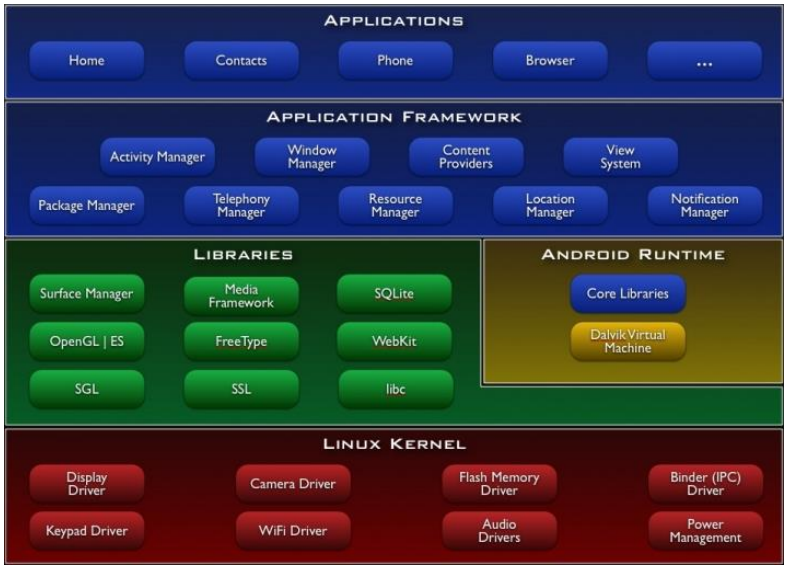
Notably, delta encoding is at the Android Application Package (APK) level only, which limits the possible reduction in patch size. To address this shortcoming and reduce update traffic even more, we extended our Delta Encoding for Less Traffic for Apps (DELTA), an update mechanism based on the bsdiff delta encoding tool.

We have implemented DELTA++ as server side software, which constructs update patches and serves them by request, and as an Android application that deploys the received patches and updates the installed applications.

**ADVANTAGES OF PROPOSED SYSTEM:**

- ✓ Unlike Google Smart Application Update, DELTA++ unpacks the A PK and then compresses its individual modules. Our experimental results show that DELTA++ can reduce application update size by 77 percent on average, relative to a 55 percent average size reduction possible with Google Smart Application Update.
- ✓ Our experiments show that additional battery use is negligible.
- ✓ Although DELTA++ is clearly superior to Google Smart Application Update in patch size and traffic reduction, its advantage in deployment time is less straightforward.

System Architecture



## IMPLEMENTATION

### 4.1**Libraries**

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

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

### Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

**Linux Kernel**

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

The Linux kernel has extensive support for and runs on many virtual machine architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate Intel x86 family of processors, though in a few cases PowerPC or ARM processors are also emulated.

**Hardware running Android**

The early feedback on developing applications for the Android platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake game The Android Dev Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.

The Android software development kit (SDK) includes a comprehensive set of development tools.[80] These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. The SDK is downloadable on the android developer website. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse (currently 3.5 or 3.6) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).[81]

**Important Android components**

An Android application consists out of the following parts:

- Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
- Views - The User interface of an Activities is build with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
- Services - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
- Content Provider - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
- Intents are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for an intent (implicit intents). For example the application could ask via an intent for a contact application. Application register themself to an intent via an IntentFilter. Intents are a powerful concept as they allow to create loosely coupled applications.
- Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.

**REFERENCES**

1. J. Wortham, "Customers Angered as iPhones Overload AT&T," 26 Sept. 2009; www.nytimes.com/2009/09/03/technology/companies/03att.html.
2. S. Musil, "Google Play Enables Smart App Updates, Conserving Batteries," *CNET News*, 16 Aug. 2012; http:// news.cnet.com/8301-1023_3-57495096-93/google-playenables-smart-app-updates-conserving-batteries/.
3. C. Percival, "Naive Differences of Executable Code," draft, 2003; www.daemonology.net/bsdiff.
4. N. Samteladze and K. Christensen, "DELTA: Delta Encoding for Less Traf_c for Apps," *Proc. IEEE Conf. Local Computer Networks*, 2012, pp. 212–215.
5. "State of the Media: Mobile Media Report Q3 2011," Nielsen, 15 Dec. 2011; www.nielsen.com/us/en/reports/2011/stateof-the-media--mobile-media-report-q3-2011.html.
6. "Cisco Visual Networking Index: Global Mobile Data Traf_c Forecast Update, 2012–2017," Cisco, 6 Feb. 2013; www.cisco.com/en/US/solutions/collateral/ns341/ns525/ ns537/ns705/ns827/white_paper_c11-520862.html.
7. "comScore Reports July 2012 US Mobile Subscriber Market Share," comScore, 4 Sept. 2012; www.comscore. com/Insights/Press_Releases/2012/9/comScore_Reports_ July_2012_US_Mobile_Subscriber_Market_Share.
8. "Background on CTIA's Semi-Annual Wireless Industry Survey," CTIA, 2012; http://_les.ctia.org/pdf/CTIA_Survey_MY_2012_Graphics-__nal.pdf.