Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Research Paper

Improved Delta++:Smartphone Application Update Reducer

Engineering

KEYWORDS :

Sarath Mathew — M.Tech Computer Science, Department of Computer Science and Engineering, Vedavyasa Institute of Technology, University of Calicut, Kerala, India

Ajayakumari K — Assistant Professor, Department of Computer Science and Engineering, Vedavyasa Institute of Technology, University of Calicut, Kerala, India

ABSTRACT

This is a compression technique that has ability to reduce the network bandwidth usage. This improved Delta++ developed under the aim of android application update compression. That means it reducing the size of updating file. When comparing with existing system like Google Smart Application Update our project achieves 50% traffic reduction. So the Delta++ can reduce cellular traffic up to 1.8% in U.S. If we can implement this type of technology in iPhone and Windows; it will reduce more network bandwidth usage.

I. INTRODUCTION

A the smart phone mean that is a phone built on a specific mobile operating system and it have many computing capability and connectivity when comparing with a feature phone. Main features of smart phones are Camera, Multitasking OS, GPS navigation, High resolution touch screens, High speed data access etc. We have many smart phone operating systems like Android, iOS, Symbian, Win8, Research In Motion. Among these android is the most accepted operating systems.

Android is a mobile operating system that developed based on Linux primarily designed for touch interactive phones. Later tablets also used these android operating systems. This OS is developed by Android inc. and Google bought this company in 2005. Apps in the Google play store increasing day by day, in 2013 apps count reached in more than 750000. Smart phone users are many now; they are using more than 675,000 applications. Number of apps Downloads per month reaches 1.5billion. The important part of an every application is the updating process; it is because of new features and bug fixes. The updates of applications may release on every weeks so it is a usual work. The following Figure.1 shows the pictorial representation of the application updates using smart phones. These updating processes result in traffic on mobile network and increase the load in data centers. Data centers are the update providers. Because of the high network traffic; mobile operators are spending huge money to upgrade their networks to keep its speed and accuracy.

High network traffic also needs more battery; so this updating process should increase the use of battery.

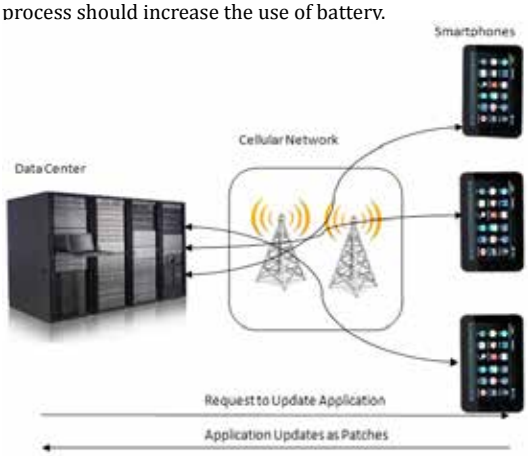

Figure.1

II. EXISTING SYSTEM

Google Smart Application Update is the new technology to reduce application update traffic which is announced by Google in June 2012. It will reduce load that induce by online app updating; that load will be high in data centers. Increased battery life of mobile devices is also promised by this technology. The technology can enable savings within the cellular networks. But this Google Smart Application performing only a single clean update process and it is not able to collect any statistics collection and accounting or any other check.

III. PROPOSED SYSTEM

In this paper a new mechanism for application update is proposed, named DELTA++. It provides substantial reduction in traffic produced by application updates. Delta encoding is a method for computing the difference or "diff", between the two files. This difference is utilized to create the advanced version from the previous version of the file. A smart phone application is updated by downloading only the difference of old version and the new one and applying the delta patch in smart phone. The key difference of DELTA++ with Google Smart Application Update is the decompression of Android APK package and compression is performed on each modules of the APK. Google's method doesn't perform this. Experimental results prove that application updates is reduced in size by 77% in average with DELTA++ whereas 55% reduction in size on average for Google Smart Application Update. Such a reduction in use of network bandwidth comes with a trade-off. When using DELTA++, more time is required to provide the application patch on the smart phone due to the increased patch complexity. This delay is accepted for updates for smart phone applications because the users don't want an update immediately when it is released. Additional use of battery is found to be negligible. Implementation of DELTA++, evaluation of the additional energy used for DELTA++, a comparison between DELTA++ and Google Smart Application Update and the estimation for large scale savings achieved by full-scale deployment of DELTA++ etc are performed in this project.

IV. THE DELTA++ METHOD

It is proved previously [4] that DELTA (Delta Encoding for Less Traffic for Apps) is useful to reduce application update traffic. DELTA is based on delta encoding tool, bsdiff [5] and it enables savings in data centres and mobile networks. A new DELTA++ method is introduced and by implementing this, further reduces the transmitted package size and attains greater savings. Size of a patch computed using delta differencing algorithm mainly depends on the total difference between two files. Use of compression in files influences resulting patch size. Suppose two files are slightly different but, their compressed versions may have great difference on binary level. This is due to the ways through which they are passed during compression. Similarly happens in the case of APK application package, too. It is basically a compressed archive of all files belonging to an Android application. The concept of DELTA++ is to calculate difference between the application files of APK instead of compressed APK packages themselves. Original DELTA method [4] produces delta difference of old version APK file of application with the new version in the form of a patch. This delta patch is generated in the server

by using bsdiff delta encoding tool. The bspatch tool is used to deploy the patch in the smart phone. DELTA works similar to Google Smart Application Update and it will not unpack the APK file. DELTA++ is improved on DELTA by exploiting the specific structure of APK package and decompressing it. Thus, patch of much smaller size will be produced in this way. The DELTA++ method is divided into two: 1) patch computation and 2) patch deployment. Patch computation is performed on the server side of the data centre. It need to be done only once for patch version of each application. Patch deployment is carried out on the user smart phone and it is repeated when an application is updated. The procedure for DELTA++ patch is followed:

1) Decompress the APK packages of old version and the new one of an application.
2) The manifest files of both are traversed to obtain names, SHA-1 hash digests and paths of each file of both APK packages.
3) The files of new version are denoted as NEW (if the file is present only in the new version, not in the old version), UPDATED (if file is in both versions but SHA-1 sums are different), SAME (if both versions contain the file but deleted in the new version) or DELETED (if old version contains the file but deleted in the new one).
4) Copy the latest version files marked as NEW into the constructed patch.
5) The latest version files marked as UPDATED are the input of bsdiff delta encoding algorithm to determine the difference of the old version with the new one. The computed difference is now copied into the constructed patch. In some cases, the difference of small files may be greater than their individual sizes. This is due to the overhead along with the creation of delta file. Here, the new file is again marked as NEW and it is copied into the patch.
6) The files marked as SAME kept untouched.
7) PatchManifest.xml file is created and it is included in the patch. It acts as a description and includes information about the application version that can be updated using this patch. It also provides information such as NEW files included in the patch and delta differences determined between UPDATED files. PatchManifest.xml file also contains information about files marked as DELETED.
8) In the final stage, the patch constructed is compressed into a ZIP archive by using bzip2. Now, the compressed patch can be sent to the Android device to be deployed in it.

Now, let's see how to deploy DELTA++ patch in the Android device.

1) Decompress the received patch into any temporary directory.
2) Application Info class is used to load APK package of current version.
3) Delete all files that are not required from the old version of application using the PatchManifest.xml file included in the patch.
4) Apply all the differences in patch to the proper files and updating them.
5) Copy all NEW files of patch into the old version of application. Now, the old version of application contains the same files of new version.
6) The APK package is created by compressing all files into ZIP archive with .apk extension.
7) Finally, the Android Package Installer, the built-in application is used to install the resulting APK package thus completing update of application.

DELTA++ has been implemented as server side software. It is responsible to construct patches and serve them by request. An Android application is developed to deploy the patches received and update the installed applications.

Table 1. Estimate of annual traffic reduction in the US

Measurement	Estimate
Number of Android smartphones [8]	60 million
Number of apps per smartphone[6]	32
Average size of an app update*	6.2 MB
Average days between updates*	29 days
App update traffic per year per phone	2.4 GB
Total app update traffic	146 PB
Total app update traffic w/ Google Smart	66 PB
Total app update traffic w/ DELTA++	34 PB
Extra savings with DELTA++	32 PB
Extra savings with DELTA++ in cellular	21 PB

* Derived from Google Play Store

V.SAVINGS OF BANDWIDTH

From Table1; you can estimate the traffic that generated due to the app updating process. The savings achieved by DELTA++ also noted in the Table 1. In 2012 United States having more than 70 million smart phone users with android operating systems. Statistics research works on smart phones shows the average size of any application among top applications in Google play is 6.2 MB. That leads to average 2.4 GB application update traffic for every user. When, taking yearly statistics updates that result in 146 PB traffic; using DELTA++ which saves 77% network traffic.

VI. RESULT AND CONCLUSION

The growing popularity of mobile devices that host multiple applications leads to significant network traffic from application updates. Compression can be a useful tool to reduce network bandwidth usage. Here we said about an improved compression method for Android application updates called improved DELTA++ that achieves an additional 50% traffic reduction when compared to Google Smart Application Update.

VII. SCOPE FOR FUTURE WORK

These projects are done only in android operating systems; If we can do it in iOS that will reduce network traffic more than android. Average size of the android apps is 6 MB while average size in the iOS is 26 MB so applying this project in apple OS will lead to significant savings.

ACKNOWLEDGMENT

The authors gratefully acknowledge the facility and support provided by Department of Computer Science, Vedavyasa Institute of Technology. Authors extend their thanks to Head of the Department Mrs. Kavitha Murugesan for her support and help during the period of project.

REFERENCE

[1] K. De Vere, "Android Reaches 25 Billion App Downloads, 675,000 Total AppsAvailable," URL:http://www.insidemobileapps.com/2012/09/26/android-reaches-25-billion-app-downloads- 675000-total-apps-available/. | | [2] J. Wortham, "Customers Angered as iPhones Overload AT&T," September26,2009.URL:http://www.nytimes.com/2009/09/03/technology/companies/03att.html. | [3] S. Musil, "Google Play Enables Smart App Updates, Conserving Batteries," CNET News, August 16, 2012. URL: http://news.cnet.com/8301-1023_3-57495096-93/google-play-enables-smart-app-updates-conservingbatteries/. | [4] N. Samteladze and K. Christensen, "DELTA: Delta Encoding for Less Traffic for Apps," Proceedings of IEEE Conference on Local Computer Networks, pp. 212-215, October 2012. | [5] C. Percival, "Naive Differences of Executable Code," draft paper dated 2003. URL: http://www.daemonology.net/bsdiff. | | [6] N.Samtealdze, DELTA Software, January 2013. URL: https://github.com/NikolaiSamteladze. | [7] "State of the Media: Mobile Media Report Q3 2011", Nielsen, December15,2011.URL:http://www.nielsen.com/us/en/reports/2011/state-of-the-media--mobile-media-report-q3-2011.html. | [8] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012 – 2017", CISCO, February 6, 2013. URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/ white_paper_c11-520862.html. | |