Optimizing Android App Update Size for Efficient Delivery

Mrs.P.Vanitha[1], Mr.N.Vinvalan[2], K.S.Aagash[3]
[1].Ph.D Research Scholar & Assistant Professor, Department of Computer Applications(UG) , Hindusthan College of Arts & Science, Coimbatore
[2]. III BCA Student, Department of Computer Applications (UG), Hindusthan College of Arts & Science, Coimbatore
[3]. III BCA Student, Department of Computer Applications (UG), Hindusthan College of Arts & Science, Coimbatore

Abstract - Android is an operating system for mobile devices, which is based on Linux Kernel, and it is designed for Smartphones, tablets and touch screen phones. Android is developed by Open Handset Alliance and led by Google. It is an open-source Operating System. There are millions of applications, which are developed in android and are released in Android Market. New versions of the existing applications are developed which overcomes the errors present in the previous application. New versions come up in market in a time span of less than week. In a newer version of existing application is developed, entire application is not changed; only a part of application which causes the error, is changed. Major part of the application is the same. Whenever the users download a newer version of the existing application, which is released in market then the entire application is downloaded. It includes re-downloading of the part of the application, which already exists in the phone. This increases the network traffic which thereby increasing the load on the cellular infrastructure. Instead of downloading the entire application again, one can only download a patch of the application, which is updated. This patch is the difference between the older version and the new version. The part of the application, which is changed, only that part can be downloaded to reduce the network traffic. In this paper, we present technique for reducing the size of the application updates on android platform by using DELTA (Delta Encoding for Less Traffic for Applications) algorithm.

Key Words: Android, Delta encoding, Delta++ encoding.

1. INTRODUCTION

Smartphone is a device which is similar to basic mobile phone but has few added features. It has more advanced computing capability than any other basic phone. Smart phone has features like touch-screen, media player, camera, GPS, web-browsing, Wi-Fi, motion, sensors, etc.

There are many mobile phone operating systems like android, iOS, Symbian, Windows, Blackberry etc. Amongst all the operating systems, Android is widely used. Android is developed by Google and it is based in Linux kernel.

According to a survey done in July 2015, there are 1.6 billion android applications available for Smartphones in Android market. Lakhs of applications are being downloaded each day from the android market. These applications are updated timely to remove errors if any and add new features. New versions of the same application may be released in a very less time gap of may be a week. Each time a new version of the application is downloaded, entire application along with the change is downloaded. This results in increasing the network traffic and brings load on cellular infrastructure. Mobile operators are spending lots of money to increase its speed and accuracy.

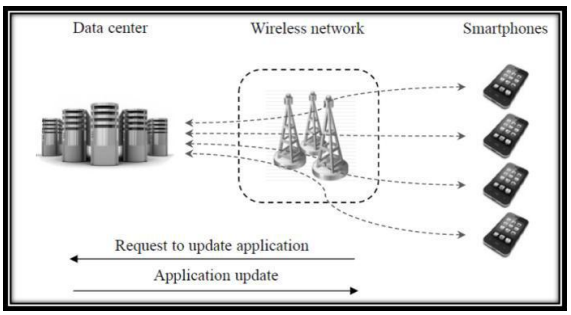

Fig -1: System view of android application updating

1.1 Distribution of Android Application

In Android, APK stands for Application Package File. APK is the file format used to install android applications on Android operating system. APK files are based on JAR file format which are used in Java

applications. APK file contains program bytecode, resources, assests, certificates and manifest files.

Whenever a new version of an application is released, its update is sent to end device. The end user downloads the new version and it gets installed in place of previous application. Simplicity is the main advantage of this method. It is very easy to download an application and install it. It requires no other operation is required. One of the major drawbacks of this method is that, even a small change is done in the new version; still the full new application has to be downloaded. Entire application is downloaded even if only few bytes are changed. This increases network traffic. Network operators are constantly spending huge amount of money to upgrade their speed and accuracy. Instead of downloading entire application, only the difference between the new application and the old application can be downloaded. This method will reduce the network traffic as the data to be downloaded is less in comparison with the earlier method. This method is known as delta encoding method. It has few shortcomings which are overcome by DELTA++. [2]

2. OVERVIEW

There are 1.6 billion Android applications. These applications may have some bugs or in some cases there is a need to add new additional features. Therefore, new versions of these applications keep on releasing. Sometimes, the time gap between releases of two versions of same application is as less as one week. When a new version of an application is being downloaded whose older version is exists in the smartphone, entire application gets downloaded. Due to this, there is an increase in traffic and load on data center. To decrease the size of the updates, Delta encoding is used. Instead of downloading the entire application again, only the difference between the two versions can be downloaded on the smartphone.

Google developed Google Smart Application Update using Delta Encoding Method. Smart Application Update was successful is downloading the difference between the two versions of the same application.

But there were few major disadvantage of Smart Application Update

1. Smart Application Update was not optimal.
2. It created data traffic instead of reducing it.
3. Its encoding was at Android Application Package (APK) level only.

Delta Encoding Algorithm was named as DELTA++. DELTA++ overcame all the disadvantages of Smart Application Update. DELTA++ is a technique used to dispatch the difference between two versions of same android application. It unpacks Android Application Package and compresses each element; hence there is a smaller patch to download. DELTA++ was further refined and named as Improved DELTA++. [4]

3. LITERATURE SURVEY

3.1 DELTA: Delta Encoding for Less Traffic for Apps

Data encoding method is also known as data differencing method. It is an approach to transfer the difference between two files rather than sending the file themselves. It takes old file and new file as input to find difference between two files. This difference is referred to as patch. This patch is the difference between the two files and patch is transferred. The implementation depends on the size of patch, amount of memory and time required to construct the patch. Two tools, UNIX bsdiff and bspatch were developed to construct patches. The bsdiff is based on bzip2 compression tool. The compression tool is an implementation of the Burrows-Wheeler algorithm. This is an open-source implementation. By taking into consideration the internal structure of the executable files, bsdiff uses suffix sorting. By considering internal structure, this method produces smaller patch than any other algorithm. Courgette is the bsdiff tool used for optimization and it is used to generate patches for Google Chrome browser. Courgette decreases the size of patch by considering the compiled application file. Disassembling is used to find all the internal pointers. By this method, it is possible to achieve 10 times smaller patch size. [1]

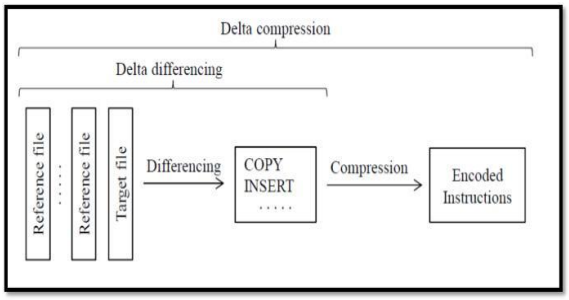

Fig -2: Delta Differencing and Compression

3.2 Delta++: Reducing the Size of Android Application Updates

Google developed Google Smart Application Update to reduce the size of the application update. It used It was improved and was used to construct a patch of the application installed on users device. These solutions were able to reduce the size of the android application update, but were not optimal. Delta encoding algorithm works on the Android Application Package (APK) level only. It limits the possible reduction of the android patch. To overcome the shortcomings of Delta Encoding method and reduce the update traffic even more, Delta was renewed and named as DELTA++. DELTA++ reduces traffic more compared to Delta encoding method. DELTA++ calculates the difference between the two files. It constructs a patch, a newer version from the older version. Content provider can update smartphone application by transferring the difference between the older version and newer version and then applying the delta patch locally on the smartphone. DELTA++ unpacks the APK and compresses each module. DELTA++ reduces size of update by 77%. DELTA++ takes more time to deploy as it requires a more complex application patch. [2]

3.3 An Algorithm for Differential File Comparison

Difference between two files is calculated by the program diff. The program diff is designed such that it would make efficient use of time and space and calculate the difference between the two files. Time and space usage depends upon the length of the file. The algorithm diff solves longest subsequence and finds the line that does not change. To obtain good performance, various techniques of hashing, presorting into equivalence classes, dynamic storage allocation and merging by binary search are used. [5]

3.4 Naive Differences of Executable Code

The android applications are updated frequently. As security laws are breached and the speeds with which they are exploited have made it necessary for the application to be updated very frequently. Binary updates are more convenient than source code updates. Distribution of pointers throughout the files makes it difficult to produce patches. Earlier methods depend upon file's internal structure to produce a patch. Naïve method produces small patches for any executable files. [6]

4. ALGORITHM

DELTA stands for Delta Encoding for Less Traffic for Applications. It is useful in reducing application update traffic. DELTA is based on delta encoding tool, bsdiff and it enables savings in data centers 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 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, this way a smaller sized patch is produced. [2]

The DELTA++ method is divided into two parts:
1) Patch Computation
2) Patch Deployment

Patch computation is performed on the server side of the data center. It needs 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.

4.1 Procedures for the Patch
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, paths and SHA-1 hash digests 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. [8]

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, this way a smaller sized patch is produced. [2]

The DELTA++ method is divided into two parts:

3) Patch Computation
4) Patch Deployment

Patch computation is performed on the server side of the data center. It needs 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.

4.2 Deploying the Patch into an Android mobile

The steps to deploy DELTA++ patch in the Android device are as follows:-

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. [8]

5. ADVANTAGES OF DELTA ENCODING

DELTA was released before Google Smart Application Update. DELTA algorithm could decrease the size of application update. It could also reduce the network traffic which saved the cellular network and data center. DELTA was improved to create DELTA++ which enables larger decrease in network traffic. Difference or diff is calculated by DELTA++ and Google Smart Application Update. Diff is the difference between two files which are

older version and the newer version. It is also known as patch. By transferring the small patch, the content provider can update a smart phone application.

The content provider can update a smart phone application merely by transferring the diff. The diff is then applied locally on the smart phone. However, DELTA++ unpacks the APK. Later, it compresses their individual module, which was not done in Google Smart Application Update.

- It could reduce the update traffic.

- It is more optimal compared to other techniques. It reduced network traffic to a greater extent compared to other techniques.

- Cost efficient. [2].

6. ANDROID APPLICATION UPDATING AND DELTA ENCODING

Android Native Development Kit (NDK) ports bspatch source code. Android 2.3.4 with API level 10 is targeted by it. There were two jobs, creating patches and applying patches. Software was developed for doing both these jobs.

The new application update process can be described as follows:

1. A patch is computed on server.
2. Patch is downloaded on Smartphone.

3. To obtain a new version of the installed application. This was done by applying the downloaded patch on the smart phone.

4. The new version of application is installed on the Smartphone

5. Patch is deleted from the Smartphone. [1]

7. CONCLUSIONS

Delta Encoding can be used in cross platform applications for decreasing the size of application updates and thus reducing mobile network traffic in Blackberry, iOS and other mobile operating systems as well. Currently rsync algorithm, which is a type of delta encoding, is used to minimize network usage. rsync is typically used to synchronize files and directories between two different systems. Similar algorithms can be used for reducing the size of application updates in different mobile operating systems.

REFERENCES

[1] Samteladze, N.; Christensen, K., DELTA: Delta encoding for less traffic for apps, Local Computer Networks (LCN), 2012 IEEE 37th Conference on, vol., no., pp.212, 215, 22-25 Oct. 2012

[2] Samteladze, Nikolai; Christensen, Ken, "DELTA++: Reducing the Size of Android Application Updates," Internet Computing, IEEE, vol.18, no.2, pp.50, 57, Mar.-Apr. 2014.3

[3] M. Burrows and D. J. Wheeler, “A block-sorting lossless data compression algorithm”, SRC Research Report 124,Digital Corporation, May 10, 1994.

[4] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula and D. Estrin, A first look at traffic on smart phones, Proceedings of the 10th Annual Conference on Internet Measurement, pp. 281-287, 2010.

[5] J. W. Hunt and M. MacIlroy, “An algorithm for differential file comparison”, scanned from Bell Laboratories Computing Science Technical Report 41 dated July 1976.

[6] C. Percival, “Naive differences of executable code”, draft paper dated 2003

[7] Shivender Singh, Anil K. Sarje, Manoj Misra “ClientSide Counter Phishing Application using Adaptive Neuro-Fuzzy Inference System” 978-0-7695-4850-0/12 © 2012 IEEE.

[8] Madhuresh Mishra, Gaurav, Anurag Jain “A Preventive Anti-Phishing Technique using Code word” International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012,4248 – 4250.

[9] Mahmoud Khonji, Youssef Iraqi, Andrew Jones “Mitigation of Spear Phishing Attacks: A Content-Based Authorship Identification Framework” 978-1-908320- 00-1/11© 2011 IEEE.

[10] Maher Aburrous, M. A. Hossain, Fadi Thabatah, Keshav Dahal “Intelligent Phishing Website Detection System using Fuzzy Techniques”.

[11] V.Shreeram, M.Suban, P.Shanthi, K.Manjula, “Antiphishing detection of phishing attacks

using Genetic Algorithm" 978-1-4244-7770-8/10/ ©2010 IEEE.

[12] Sadia Afroz, Rachel Greenstadt "PhishZoo: Detecting Phishing Websites By Looking at Them".

[13] Michael Atighetchi, Partha Pal "Attribute-based Prevention of Phishing Attacks" 978-1-4673-2104-4/12/ ©2012 IEEE.

[14] Huajun Huang, Junshan Tan, Lingxi Liu "Countermeasure Techniques for Deceptive Phishing Attack" 978-0-7695-3687-3/09 © 2009 IEEE.

[15] Shinta Nakayama and Hiroshi Yoshiura, Isao Echizen "Preventing False Positives in Content-Based Phishing Detection" 978-0-7695-3762-7/09 $26.00 © 2009 IEEE