Unwind: Mobile Data Offloading

Bharadvaj Jani, Mithilesh Gholam, Tanumay Medda, Vishakha Shelke
BE Student, BE Student, BE Student, Asst. Professor
Dept. of Computer Engineering,
UCOE, Vasai, India.

Abstract : Although the advancements in mobile technology are reaching heights, the computational problems still persist degrading the performance . The low capacity batteries coupled with the high end processors and the capabilities limit it's usability. Thus, Mobile Cloud Computing (MCC) combines mobile computing and cloud computing in order to improvise abilities of mobile devices using offloading. Data offloading is the transferring of resource concentrated computational activities to other platform like cloud. This paper proposes an android application to offload mobile data to cloud in order to have a backup of data, also features local and remote offloading. The reported experimental results show effectiveness of our solution when applied to low storage capacity device ,offloading user defined data to cloud or external platform with ease by data offloading methodology.

IndexTerms **- Mobile computation, Cloud computing(CC), Mobile Data Offloading.**

I. INTRODUCTION

Now-a-days, mobile application are expanding themselves in terms of functions and features to give higher usage capabilities that challenges execution time and usage of energy. The smartphones developed at the early era of modernization in android market where designed with small storage bars and low profile processors which where able to handle the data then. Today with expanded android market with too many developers catering to users with different application to make their lives easy complicate things on the storage side as users have to choose what data to keep and what to get rid off.

UNWIND allows moving personal data from devices running short of resources to secondary platform; it can lower energy consumption at the expense of moving calculated data to offloadable servers. In other words, it is understood as, dealing with important data on a mobile device is prone to accidents hence to offload to secure platforms to cure the problem of data loss.

UNWIND will perform a task selection algorithm which indeed will make ask the user for choice of offloading data as Media comprising of images, videos and audio files and application list for which it will search for all the present application devices and sort them according to their storage size. Other than that it will also create a list of important data members of the mobile device like contacts, calendar data, alarm ,etc.

Therefore, to maximize usability of our system and to maximize the benefits of offloading, an easy and systematic task selection algorithm is needed with a minimalistic user interface design. This paper specifically presents the design and implementation of application based on Mobile Computational Offloading Framework and ULOOF framework that will help user to create backups of important data at various instance with ease. This method has been implemented and tested on a local android mobile device to test the backup and recovery capabilities indulging cloud service and data sharing over IP connection using WiFi network API.

II. LITERATURE SURVEY

The following research articles are selected for review, keeping in mind the traditional and conventional approaches of our application :

K. Kumar and Yung-Hsiang Lu worked on Mobile computation offloading [1] allows dynamically migrating computation-intensive applications from resource limited devices to external machines in their "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" [1].

P. Bellavista, A. Corradi, A. Zanni worked on modern mobile computation offloading technologies that should dynamically consider the opportunities associated with fog or mobile edge nodes, in addition to the ones targeted by traditional mobile computation offloading, in order to go beyond the sole direct interaction of mobile devices and global cloud datacenters in their "Integrating Mobile Internet of Things and Cloud Computing towards Scalability: Lessons Learned from Existing Fog Computing Architectures and Solutions" [2].

F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, proposes in their work about Fog computing consists of Edge nodes directly performing physical input and output often to achieve sensor input, display output, or full closed loop process control in "Fog computing and its role in the internet of things." ACM Press, 2012 [3].

Beck, M. T., Werner, M., Feld, S., and Schimper, T suggested that MEC technology is designed to be implemented at the cellular base stations or other edge nodes, and enables flexible and rapid deployment of new applications and services for customers in Mobile edge computing: A taxonomy, in Proc. of AFIN 2014. [4]

A. V. Kempen, T. Crivat, B. Trubert, D. Roy, and G. Pierre stated that Mobile applications can benefit from computation capability of fog and MEC nodes to remain available and responsive while processing large volumes of data in 'MEC-ConPaaS: An experimental single-board based mobile edge cloud" [5].

S. Yi, C. Li, and Q. Li, suggested One of the main, still open, is the technical challenges related to mobile offloading is how to automatically partition an application code into offloadable and non-offloadable parts, for arbitrary applications by learning application code architecture in "Proc. of ACM Mobidata 2015".

III. PROPOSED SYSTEM

This data selection algorithm takes help of android application program interfaces to check the included methods that are present on the device. The proposed system is built to be generally-adaptable and independent from app-genre, sizes, structures and domain. In particular, this application triggers the mobile data offloadation framework to deliver backup data at remote location at end user level. To maximize fault tolerance data to be offloaded is categorized according to type shown in following table 1

Types of checks	Classes	Methods
Internal folders on local storage	yes	yes
Class and data file offloadability	yes	yes
Internal method call	No	yes

Table 1

It is important to segregate the data into such categories as not every file is offloadable as internal calling method which would be responsible for triggering the execution would be offloaded too, e.g., non-serializable Java objects. All segregated data i.e. Android process files and other files in img, jpg, etc format that are involved into a non-offloadable class table are termed as non-offloadable. Similarly, an offloadable data is a data that passes all the class checks serial-wise and whose methods would be considered as possible offloading options. The android xml is designed in a way to minimalize the complexity to end user. The following fig shows the Mobile Computation Offloading Framework.

3.1 Configuration File Loading

Figure 1 shows the Mobile Augmentation Cloud Service which bridges the Application system to cloud services architecture. The use of MACS middleware in application benefit in seamless offloading of computation-intensive parts of the application into nearby or remote clouds. According to different conditions/parameters on the class files of android application, the modules of program are divided into two groups; one group runs locally to compute size and space of the data, the other group is utilized to export the offloadation service on the cloud side.

The advantage of having a backup of data on cloud is even sometimes the hardware based backup is lost if hardware is damage and even the inconvenience is cut by using a Cloud based backup service. Here Unwind supports three different cloud platforms for data migration namely Dropbox, Google Drive and Reware cloud. This helps in seamless access of data over any device while data is backed up only from one end .

3.2 Unwind-Android Application: Internal Structure

Internal structure shows various modules implemented to backup data via different sources and through different platforms.

3.2.1 New Backup:

There are two sectional views:

3.2.1.1 Local Backup:

This sectional module enlists installed application on the mobile device using Application Program Interfaces. The function of local backup trigger s the subroutine directory along with main file in the called function and the directory location of the storage center. The Activity file is parsed locally and combined with main manifest details to compress a zipped file at the desired local address ,i.e apk file on external storage.

3.2.1.2 Cloud based Backup:

The module triggers the MACS middleware along with ULOOF framework and fetches the data from the device itself and the sorting method used gives a segregated classification of data in the form of installed application and media files according to their respective extensions. It uses a set of tests to detect which files are not suitable to be offloaded, as

File Size, local execution time, and file location. This method generates the list to choose from and generates the connection requests from MACS library used in Java to Remote Execution center where the requests are scheduled with a pipeline like structure on request and retrieve basis, when a file is selected and tested then the remote execution manager decides to fire the connection with the cloud service and establish a secure connection and offloading takes place. This data is scheduled in encrypted format while transaction with synchronization points.

Synchronization points are buffered memory collectors which store current transition offloading details and helps to retrieve when failure arise. The successful offloadation toasts on the users screen with a positive feedback.

Figure 2

3.2.2 My Backup

The offloaded data to the cloud replicates itself in this module. The service manager on MACS framework takes the service from user and checks the connection with cloud server, on affirmation, the data present on the cloud is listed in the categories segregated before hand of offloading with specified nametag to make it easier to manage.

3.2.3 Migrate

In this module , two devices are required to be on the same network so as to migrate the data files from one node to another. This here is achieved with the held of NSD(Network Service Discovery) and uses Wifi (peer-to-peer). API automatically searches for devices connected on same Wifi network ,i.e. if connects then toast else error message generation and after connection the framework handles the service similar to the cloud based backup.

3.2.4 Schedule

To make the backup at specific time, schedule module is embedded with the alarm API of android and the alarm function is synchronized with the offloadation module of new backup which works exactly like the alarm clock and the satisfactory condition triggers offloading. This gives user a freedom to offload important data files at the least active hours by scheduling it.

IV. RESULTS AND DISCUSSION

4.1 Result:

This paper proposes an innovative way that autonomously parses a mobile Android app data and retrieves the list of methods that are suitable to be offloaded to a cloud/edge node or local storage using cross platform cloud support. The reported experimental results show that our application can create an instant backup of data according to user's convenience. In addition, our application has demonstrated to be able to perform a complete scan on large-scale real-world apps available on the device which are downloaded from google play store. These results are stimulating our ongoing research work in the field of cloud computing and extension of the Mobile Data Offloading framework and helping users to offload data over cloud without loosing a bit and access from different devices with a simple login.

Figures

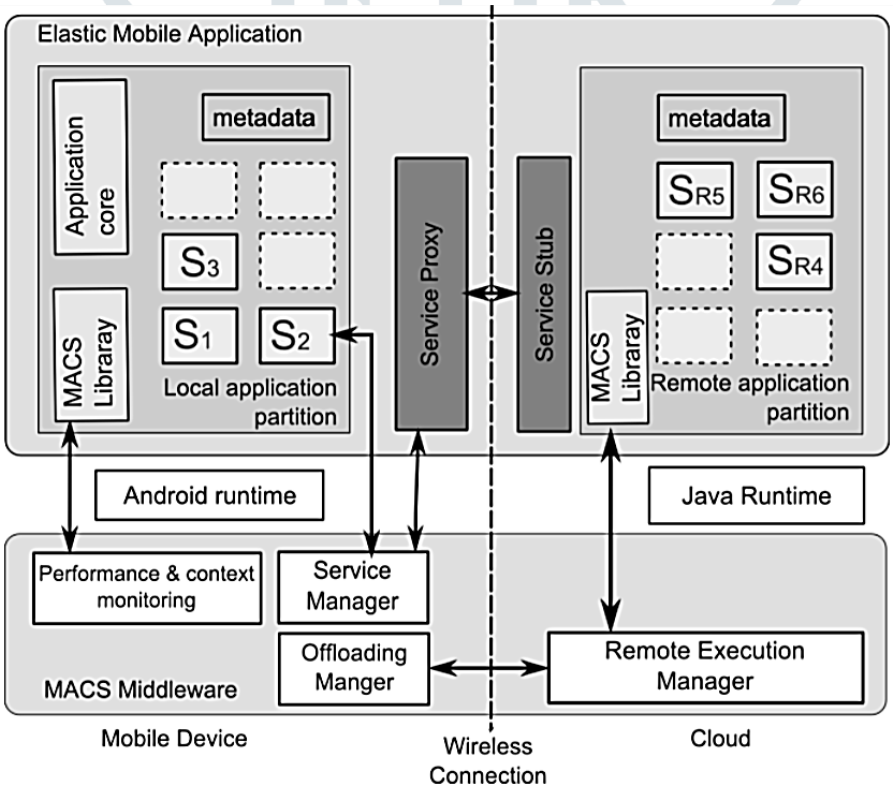

Fig.1

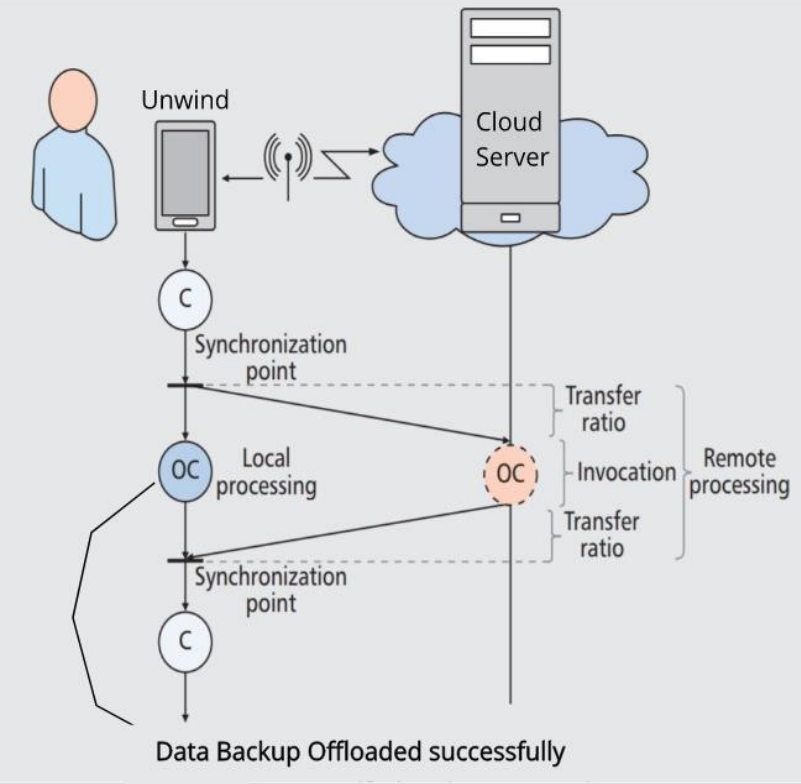

Fig.2

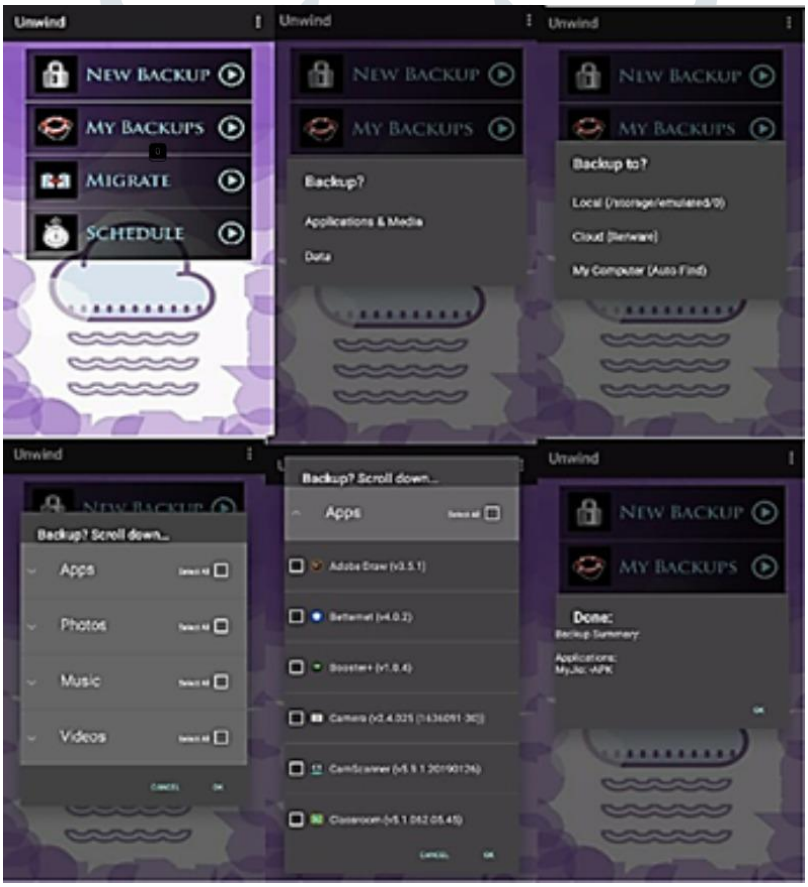

Fig.3

Figure 3 shows the applicational overview of the system at end user's device.

First Selection of a module from the four modules present then Backup media then choosing platform for storage of backup and finally login to cloud if backup media is selected as cloud and the data will be backed up categorized as per selection.

IV. REFERENCE

[1] K. Kumar and Yung-Hsiang Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" Computer, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[2] P. Bellavista, A. Corradi, A. Zanni, "Integrating Mobile Internet of Things and Cloud Computing towards Scalability: Lessons Learned from Existing Fog Computing Architectures and Solutions," ser. 3rd International IBM Cloud Academy Conference (ICA CON) '15, 2015.
[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things." ACM Press, 2012, p. 13.
[4] Beck, M. T., Werner, M., Feld, S., and Schimper, T. (2014). Mobile edge computing: A taxonomy, in Proc. of AFIN 2014.
[5] A. V. Kempen, T. Crivat, B. Trubert, D. Roy, and G. Pierre,
"MEC-ConPaaS: An experimental single-board based mobile edge cloud," Apr. 2017.
[6] S. Yi, C. Li, and Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues," in Proc. of ACM Mobidata 2015