Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 7 (2017) pp. 2037-2060

http://www.ripublication.com

A Study on the Techniques of Computational Offloading from Mobile Devices to Cloud

Abhishek Bajpai
Dept. of Computer Application, SRMU, Barabanki-UP, India.

Shivangi Nigam
Dept. of Computer Application, SRMU, Barabanki-UP, India.

Abstract

In the current scenario, there is an intensive use of smart phones by the mobile users and with the emerging Mobile Cloud Computing (MCC) concept the issue of offloading from resource limited mobile to resource rich cloud comes into existence. The execution of various applications and codes in the smart phones is battery consuming and the performance of the smart phone degrades due to this. As the smart phones have limited battery life and low data storage capacity, the offloading from mobile to cloud helps to overcome these problems and the performance, in terms of the computational efficiency, of the device can be improved. This paper presents a survey of the research studies on the offloading in MCC. In this research we are going to discuss the various techniques of computational offloading, their advantages and issues associated it. The paper also discusses related future research directions in the related context.

Keywords- Cloud. Smart phones. Cloudlet. Clone-Cloud. Mobile Cloud Computing (MCC). Offloading, Energy-Efficiency.

INTRODUCTION

The smart mobile phones have become an integral part of our lives and make our day-to-day work easier. The smart phones have the features like GPS navigation, web browsing, Wi-Fi etc., that could be used anytime and anywhere. Despite of all these advance features, the major issues that we face with reference to these smart phones are limited battery life, low data storage capacity, limited bandwidth etc. The MCC has proved to be a promising solution to tackle the problems associated with the smart

phones [1]. It is a technique to connect mobile devices to a cloud, with vast resources in terms of hardware as well as software, for the purpose of computations related to various mobile applications, processing, storage of data etc. [10, 14]. The mobile cloud that have unlimited resources virtually run the applications for the smart phones to save the battery, to improve the performance and to improve the overall computation efficiency. In recent years, the researchers have greatly focussed on the field of MCC. The vast domain of the applications of MCC attracted the researchers to devote their time in it and make their contribution; due to this the field of MCC has been growing so rapidly. Figure-1 gives the broader view of MCC.

The area of MCC is so vast so it has a number of applications in the field of mobile healthcare, mobile gaming, online business transactions, e-learning, multimedia services etc. [6,28]. There are several issues related to MCC like limited bandwidth, service unavailability due to network congestion or may be due to link failure, heterogeneity of the networks [15, 29] that access the services of the cloud, privacy, security etc. But the scenario of the computational offloading from smart devices to the cloud is the area of concern in this paper. The computational offloading is the method to offload a fragment of code to a cloud for executing it remotely in order to save the battery life [23,42]. Offloading form mobile to cloud plays an important role in exploiting cloud services to improve the performance of the smart phone.

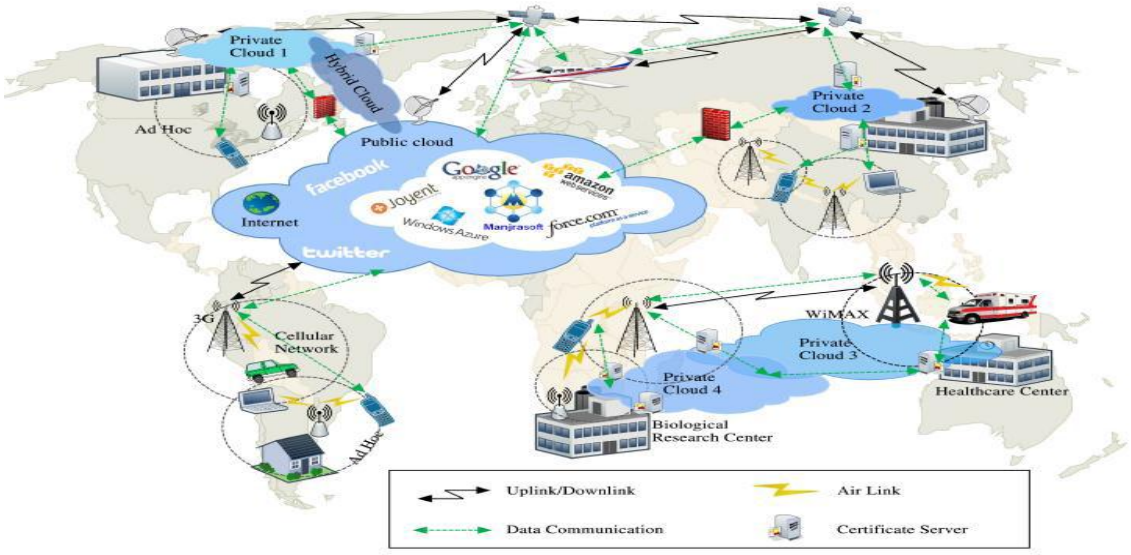


Figure 1: MCC: Conceptual View

There are several applications and codes that need various software platforms or tools for their execution that may not be installed in the smart phone due to its limited memory capacity, for using them the offloading is mandatory to cloud which is rich in resources [16]. As well as the any situation may occur in which the data back-up is required, so in that too the offloading plays an important role and a back-up has been created at the cloud. Now, the most critical question that arises at the time of offloading is what could be offloaded or what could not [37]. What part of data or

application should be offloaded from the mobile phone to cloud in order to maximise the energy efficiency with optimised cost, how the code need to be partitioned etc. This paper focuses on the survey of various techniques and models proposed for offloading to the remote server, its benefits, applications on the basis of its complexity, energy saving, security etc. A detailed survey of computational offloading is presented here. Section-II shows the overview of the computational offloading. In section-III discusses the techniques of offloading. Section-IV provides the future research direction in the area of concern. Section-V gives the conclusion of the overall survey paper.

OVERVIEW OF OFFLOADING

The concept of MCC involves the offloading of the task that is to be executed by the remote server. The part of application that need to be offloaded from mobile phone to cloud could be done in two fashions that is partial offloading or full offloading [20,41]. In the full offloading architecture the full application along with all the data associated to it has been offloaded to the cloud where the entire computation take place and the final results has been sent back to the mobile device as shown in figure-2.

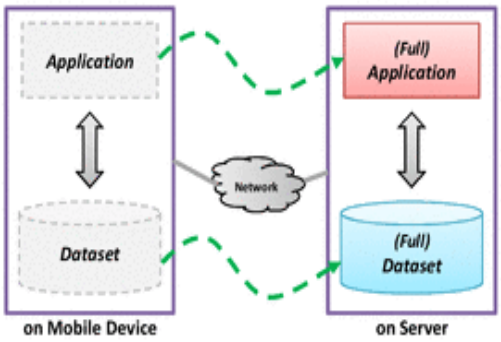


Figure 2. Mobile offloading

In the partial offloading architecture only the part of the application that consumes more energy or have complexity in terms of computation has been offloaded to the cloud. In this both mobile phone and the cloud are responsible for the computation and final results comes after merging the individual results of both the computations that is in mobile device and at the cloud as shown in figure-3. The partial offloading computes task locally, which could have been complex due to lack of resources in terms of software, infrastructure etc. Moreover, the runtime complexities could also be avoided if the code is offloaded to the cloud for execution.

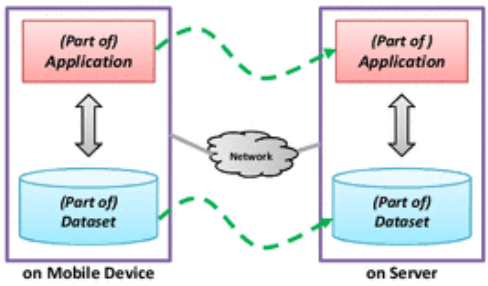


Figure 3: Scenario of Partial Offloading

Backup of Smartphone User's Data:
The smart devices have limited memory and storage. Also, it may face some technical failure like software crashes etc. In that case, user can lose the valuable data that is saved to the cloud. So, to avoid all these possibilities a data backup could be created at the cloud and hence, the user can retrieve his data anytime from anywhere.

The major benefits of offloading that has been observed through various researches are:

A. Benefits of Offloading

Improvement in Lifespan of the Battery of Smart Device:
If all applications would have executed locally by the mobile device then it would consume a lot of power of the mobile device to accomplish the whole task. As the mobile device is energy constraint therefore, to save this power some part or the complete application code is offloaded to the remote server and it is then get executed by the cloud and only the final result is sent back to the mobile device [6,26,35,43]. So, by this methodology the energy consumption could be lessened for the same.

Code Partitioning:
The codes of the apps need to be offloaded to the remote server where it is being executed in order to save the communication cost. Now, the question arises that what part of the code is to be offloaded and what part should be executed locally by the mobile device. The code partitioning could be done in two ways as shown in figure-4 that is static and dynamic.

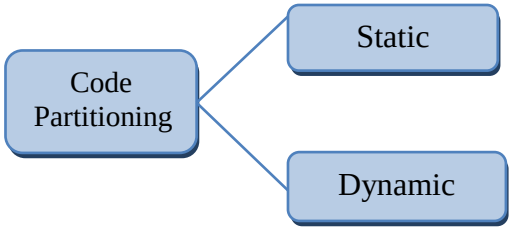


Figure 4: Methods of Code Partitioning

In static code partitioning method, the part of the code to be offloaded to the remote server is pre-set by the mobile devices. But, this method is a failure in today's heterogeneous network environment. So, to overcome this problem, dynamic code partitioning comes into existence. In dynamic code partitioning, what part need to be offloaded to the cloud has been decided at the run time depending on the available resources, complexity of the code etc. [4, 8].

Enhancement of Performance of Smart Device:

B. Applications of Offloading

Cloud Assisted Exploration

The offloading the data to the cloud might be helpful for global cloud-assisted search. If a user uses more than one mobile device in the past then to retrieve the data from previously used devices could be easily done by exploiting the cloud for the global search [24,45]. This application of offloading is highly time-saving provided all the devices are synchronized their data with the cloud and indexes have been updated.

Mobile-Gaming:

The gaming applications consume high power of smart device and require high memory usage. So, the mobile gaming utilises the services provided by the cloud and it enhances the gaming experience by providing better performance as the gaming application will be executed at the cloud.

The transparency will remain intact at the user end with respect to the execution of the game application that whether it being done by local mobile device or the remote server.

Educational Tool:

The users can exploit the services provided by cloud by their smart devices for the educational purposes [9]. The tools have been developed that exploits the cloud resources for the processing of multimedia files. With the help of this the students can access various real time applications and algorithms for educational purposes.

Health Care:

The excessive use of smart devices by the users keeps them connected with the cloud almost all the time. So, this could be helpful in users' health monitoring and care [13,27,33]. The various sensors could be attached to the body of the user for the monitoring of blood pressure, pulse rate etc. This data could be offloaded to the remote server at certain time intervals by the nearest mobile device and in case of emergency health of the users.

Cloud-Assistive Vehicular Transportation:

The automobiles are now-a-days fitted with modern gadgets with internet connectivity. This could lead to the emergence of intelligent transportation by exploiting the services of the MCC [12]. The automobiles are now-a-days comprises of sensors like proximity sensors, temperature sensors etc. And, with the help of

environment sensing the remote server could help in managing the traffic, safe navigation etc. This may also be used for inter vehicular communication etc.

Robotic System Designing:
The robots also suffer with the problem of high energy consumption and low storage capabilities. So, merging it with the concept of MCC will increase the efficiency of the whole System [17].Some features could be incorporated in robotics system that will help the robots to communicate with each other or with the cloud.
So, the less power will be consumed and the resources and the data could be shared among the robots present with the cloud.

Forensic Applications:
The fast growing era of digital forensics enforces the scientist to merge it with the MCC [25].This would help in retrieving the information and data stored in the smart devices for the purpose of investigation and surveillance.

Mobile Calendar Prediction:
The mobile users make use of intelligent calendar system for scheduling the day to day work with the help of cloud services [30]. This helps in scheduling the work by comparing it with the work already done and is saved at the cloud.
The link of cloud with intelligent calendar system helps in providing the changes in the user's usual activities if observed using accelerometer and can predict or recommend the activities the use may perform during the day.

III. MODELS OF OFFLOADING

The high energy consumption is the severe problem in smart phones which is to be improved. The problem of energy consumption could be tackled by offloading from mobile to clouds and hence executing the problems at the remote server. In this section various models and architectures of computational offloading are explained [44].

A. MAUI Architecture

The architecture of MAUI provides a framework for offloading from mobile device to smart phone for the sake of energy saving [5]. In this architecture first of all, it has been detected that whether the code should be offloaded or it should be executed locally, the MAUI profiler, as shown in the architecture figure-5, is responsible for this. The decisions of offloading is taken after considering the factors like mobile phone's pattern of energy consumption, running time of programs, available bandwidth etc.

After this, the collected data of profiler is used up by the MAUI solver which gives the information that what is executed locally and what should be executed at the cloud and picks that code partitioning method in which the energy saving by mobile could be maximised. Here, there is also the provision of handling the failures that may occur due to the factors like the network failure, device malfunctioning etc. Here the failure

detection has been done on the basis of timeout mechanism. At the time of connection failure the MAUI gives the control back to the local proxy that could either restart the process locally or it may find an alternate MAUI server and then restart the process. The proposed can work efficiently and saves the energy of smart phones considerably.

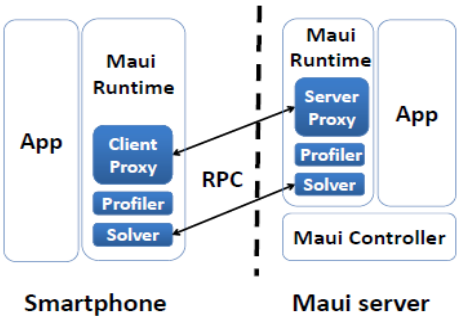


Figure 5: Architecture of MAUI

B. Clone- Cloud Based Models

The clone cloud system [3] has been developed for the offloading from mobile device to cloud automatically; unlike MAUI it doesn't require programmer's assistance. It involves execution offloading in which offloading of a process takes place during the runtime. Clone cloud involves dynamic code partitioning, as shown in figure-6, and state migration. At the runtime the code has been partitioned because the resource availability is known at that time and hence better decisions could be taken to optimize the migration cost and energy consumption.

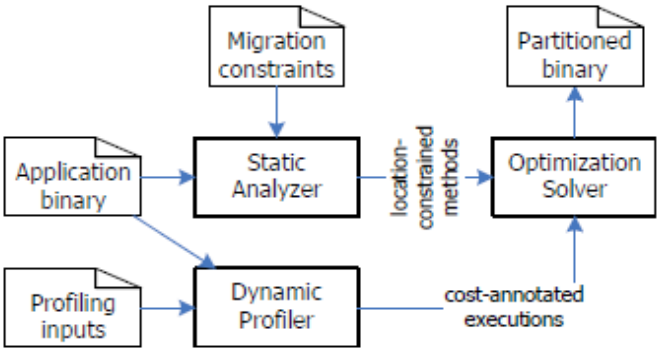


Figure-6: Dynamic Code Partitioning Framework

In the clone cloud system, code partitioning has been done in the same way as in MAUI that involves profiler and solver except the fact that the working of the migration of clone cloud is on the basis of granularity of the thread. The steps involved in the thread migration are as shown in figure-7. First, the thread migrator involves the processes like suspend, capture, resume and merge. The thread migrator

suspends the migratory thread and all of its states have been passed to the node manager. After this, a state of the thread has been captured on the basis of its various data sets. The captured thread state is offloaded to the clone cloud. For the reverse path from the clone cloud to mobile the capture mechanism must be reverted and the thread is resumed. In state merging, context gets updated for the thread with all effects happened at the clone and hence the thread is reintegrated in the mobile.
Second, the responsibility of node manager is observed which involves the communication between mobile and clone cloud and is responsible for the migration of packaged thread. Third, a partition database has been created in order to check which partition should be used so as to minimize the energy consumption. This model is very efficient and save considerable energy of the smart phone along with the optimised cost.

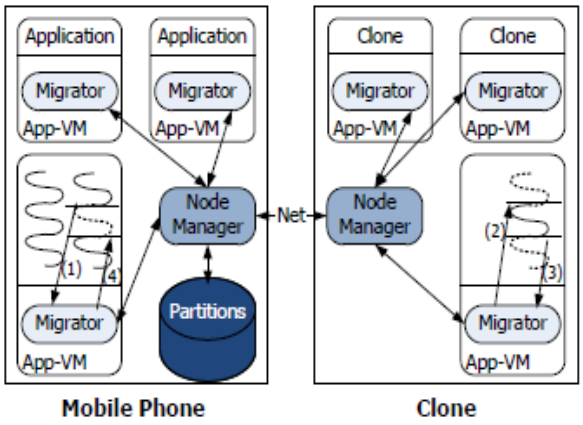


Figure 7: Migration Process

The total size of the objects offloaded governs the transfer time over the channel. The clone cloud system transfers the complete thread from mobile to cloud; the fast execution offloading [34,46] overcomes this problem. In this model, instead of transferring all the RHOs (Reachable Heap Objects) of a thread only the EHOs (Essential Heap
Objects) have been transferred over the network at any migration point. Hence, the size is reduced considerably and success rate increases. Only a few RHOs of a thread are required at the run time, so all of them need not to be migrated. Whereas on the other hand, EHOs are transferred that have specific references in a thread. Hence this model further improves the performance and gives an optimised solution in reference to the cost and energy consumption.
Another architecture has been developed in which both the offloading and creating the data backup at the remote server has been proposed [2]. Here, two types of clone clouds have been explained: off-clone which is responsible for the offloading from smart phone to the cloud and back-clone is responsible for the data/software back-up of the Smartphone's data at the cloud. The mobile's activities need to be monitored to study the overheads, to accomplish this, the Logger application has been developed

for the android phone. The logger logs all the events of the smartphone. Firstly, passive data is collected. For this, the Logger considers Intents to acknowledge for the connectivity status, screen mode, battery usage etc. The gathering of active data is also required. For this the logger sets some alarms that logs the data exchanged across the network interface. The android OS already has I-notify that includes watches, which logs the file system activities. When I-notify send the information about any change in the file system activity, the amount of data to be offloaded to the cloud needs to be decided. For this purpose, Roling-Hash technique is used.

This technique quickly computes the data to be uploaded to the cloud; the selected data should be fast and lightweight. In this architecture, collection of information by all the components is repeatedly saved in a log file. The file is then compressed and stored in a directory in the SD card. It has been observed that the off-clone requires the perfect synchronization with the mobile so it increases the energy overhead as compared to the back-clone that requires less network traffic.

Differences between MAUI and Clone-Cloud Models

MAUI architecture and clone cloud framework both are the efficient techniques of offloading that optimizes the energy consumption by the device. But the main differences between the two techniques [18] are as follows:

In MAUI system, the presence of application binaries are required both at the smartphone and at the cloud whereas in clone cloud architecture, the device clone is there in the cloud, application binaries and partitioning databases are present in the smartphone.

The method granularity is RPC-like in MAUI and thread suspends and resume-like in clone cloud.

The availability of source is required for MAUI which is not a constraint for clone cloud.

MAUI focuses on maximising the energy saving whereas clone cloud focuses on minimising the execution time and energy consumption.

In case of MAUI, UI codes, codes that have I/O involvement and the codes that have local execution time less than a predefined value could not be migrated to cloud. As far as clone cloud is concerned, mobile-specific methods, methods that share native states and nested migrations are not feasible.

MAUI's application model is annotated call graph and that of clone clouds is annotated profile tree.

MAUI's implementation is based on .NET (compact) model and clone cloud is implemented based on the Java Dalvik Virtual Machine.

Table I: Remote Execution Unit Of MAUI Architecture And Clone-Cloud Based Model

Method	Remote Execution Unit
MAUI Architecture	Methods(RMI)
Clone-Cloud Based Model	Threads

C. Cuckoo Design

The cuckoo design [19] has been proposed for offloading from smartphones that run on android platform to the cloud. This model supports both local and on- cloud execution of the code depending on the available resources. This model works on an already existing activity/service model in android. The cuckoo components have been integrated into the android builder process. First the Rewriter writes the stub for all AIDL interfaces. It is done so that at compile time it could be decided that whether the execution should be done locally or on-cloud server. Second, the Remote Service Driver derives the implementation of dummy remote which is done by the programmer. Ant file is also generated which then generates the jar file that is executable at the cloud. After that, it is decided that whether the execution should be done locally or at remote server, the cuckoo resource manager monitors the reachable resources and provide one if possible. Any system that uses cuckoo model could offload to remote server that has java VM.

D. MACS Architecture

In the Cuckoo system, static code partitioning is done at the compile time. Another system, Mobile
Augmented Cloud System (MACS) [21] has been proposed which is similar to Cuckoo system but it involves dynamic code partitioning at the run time. The system monitors the available resources and it then draws an optimised solution that whether the offloading should take place or not. In this application could run on the android phone but could also be offloaded to the cloud on demand basis. The MACS model is as shown in figure- 8. If any application need to use the cloud services could send their query to the service manager then the service proxy is created.

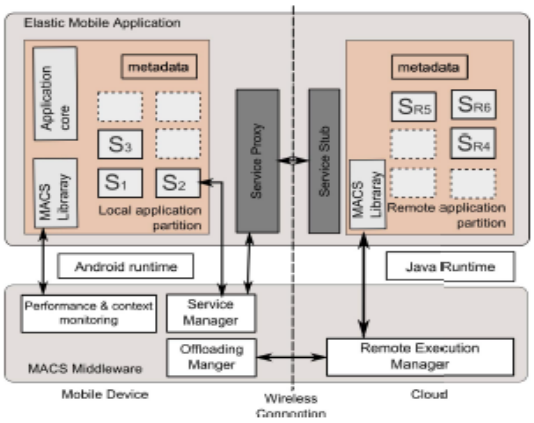


Figure 8: MACS Model

After the execution the results are returned to the android phone via service proxy. The system focuses on achieving three factors for successful and efficient offloading which involves that memory usage should be minimum, energy consumed should be minimum and the execution time should be minimum. The MACS framework consumes less energy even for complex processes.

E. AHP and TOPSIS Techniques based Design

In the present scenario, there is the availability of many similar kinds of clouds, so it is up to the smartphone to choose one of them for the purpose of offloading depending on the parameters like cost, availability, bandwidth etc. There is a technique that was proposed for cloud-path selection [32], which decides which cloud should be chosen to offload on the basis of analytical hierarchy process (AHP) and technique for order preference by similarity to ideal solution (TOPSIS). The decision hierarchy is as shown in figure-9. Here, AHP considers multi-criteria problem and it analyses and find the relative weight of all the parameters to be considered for cloud selection availability, security, speed, price and bandwidth.

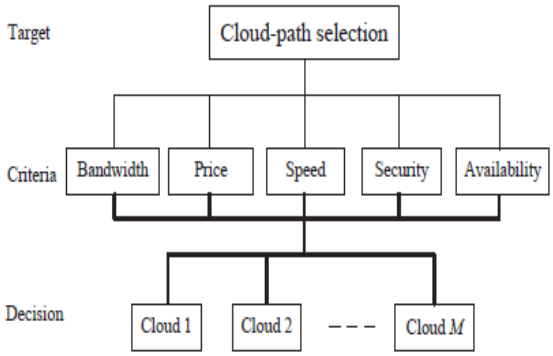


Figure-9: The Hierarchy to Decide Cloud Selection

The fuzzy TOPSIS technique is used here to find the final ranking of the clouds. The decision making has been done by considering fuzzy approach. It has been observed that if the data privacy is not the prime concern then the parameter of security doesn't play the important role while selecting the cloud. Speed is an important factor because it governs the execution time. Bandwidth should also be considered to modify the communication cost. Here, the parameters have been prioritised as: Bandwidth>Speed>Availability >Security>Price

F. Energy Aware Design for Workflows

In order to improve the application performance and to save energy, an approach has been presented in which the mobile workflow has been offloaded to the nearest available cloudlet [11], a scenario is shown in the figure- 10. In this, first it is been decided at smartphone end that which cloudlet should be chosen. The best offload nodes are selected depending upon various real time attributes. Then, all the tasks have been observed serially and decision need to be taken that whether there is the requirement of offloading or not. In case of the need to offload, the workflow engine forms the cluster tree. And the job is considered as the root node. The task clustering is done in order to save the energy required for communication. If two or more tasks have been offloaded to the same cloudlet for if the workflow is being spread on one or more than one smartphones then it could save energy as only one well established route could be considered again and again. Here, there is also a provision of offload authorization by the owner of the smartphone. The owner may or may not let the offload to happen depending on the privacy of the data. Here, it has been observed that the offloading is affected by two set of parameters that are:

•Communication Size and Network Connectivity
•Computation Size and Cloudlet Processing Speed

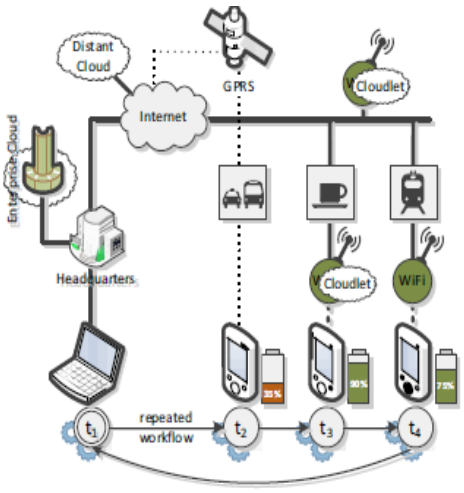


Figure-10: A Scenario Showing Cloudlet and Different Networks to Save Energy of Smartphones

G. MCSOS Architecture

A framework like previous one has been proposed for offloading from smartphones to nearby cloudlets to optimise the energy consumption called mobile cloud with smart offloading system (MCSOS) [32], as shown in figure-11.

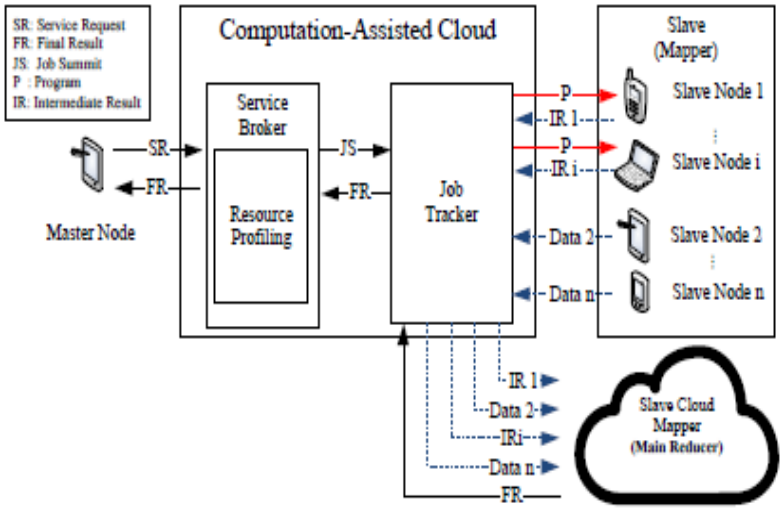


Figure 11: MCSOS Model

The MCSOS framework exploits Map Reduce programming model for optimizing the energy consumption. For getting the better performance, the master node i.e. smartphone, forwards the task to the service broker which then divides it among the slave nodes around the master node for the purpose of parallel computation of job. The slave nodes could run the task locally or it could offload it to computation-associated cloud (CA-cloud) depending on the resources. The results drawn by the slave nodes is then submitted to the CA-cloud where all the results are integrated and a final result is sent back to the master node via job tracker. In this, three types of situations have been tracker.

In this, three types of situations have been analysed. First is the one where no offloading is done, as shown in figure-12, that is all the slave nodes locally performs the job and final results of all the nodes have been integrated at the CA-cloud.
The second scenario considers full offloading as shown in figure-13, in this the mappers task of all the slave nodes have been offloaded to the CA-cloud and all the execution is then done at the cloud and final result is sent back to the master node.

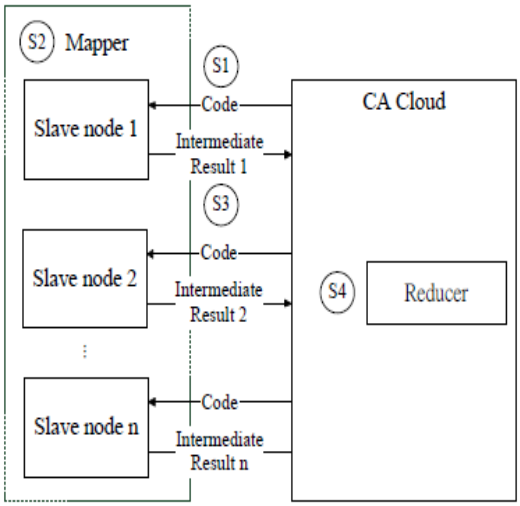


Figure-12: No Offloading Scenario

Third is the hybrid offloading scenario, which is the combination of the first and second, where decision need to be made that whether the offloading should be done locally or a CA-cloud depending on the available resources at the slave node and distance from the cloud. The MCSOS framework works on hybrid-offloading scenario as it gives the best optimized results and minimizes the energy consumption.

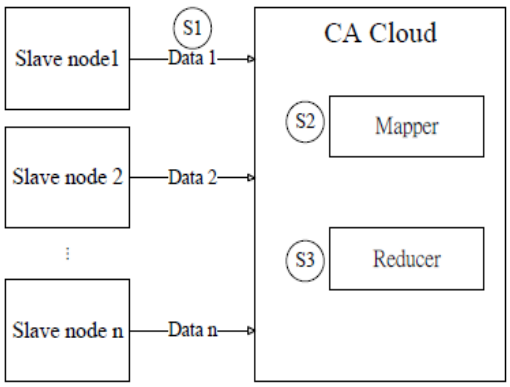


Figure-13: Full Offloading Scenario

A model has been proposed for applications associated with the mobile health care [27]. The sensors are used for the health monitoring but the multimedia sensor signal processing requires the algorithms that have high complexity, requires huge memory space for the storage and consumes much power so it need to be offloaded the cloud. The proposed model considers three components in the system that is body sensors, mobile device and the cloud, the architecture is as shown in the figure-14.

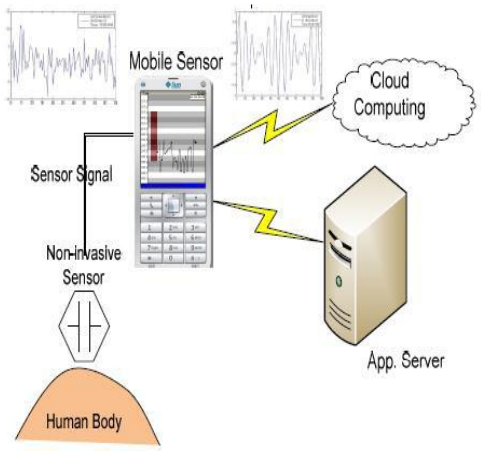


Figure 14: Mobile Health Care Monitoring Architecture

The system classifies the computational process in two classes that are weak and strong classes. Weak class processes are those that could be executed by the smartphone due to low complexity, low power consumption, low security etc. But the strong class processes need to be executed by the cloud on transitional basis that is main execution is done in the cloud and the end results are then further refined and completed by the smartphone.

The model for mobile healthcare applications has been further explained for next generation networks like 3G or 4G as shown in figure-15.

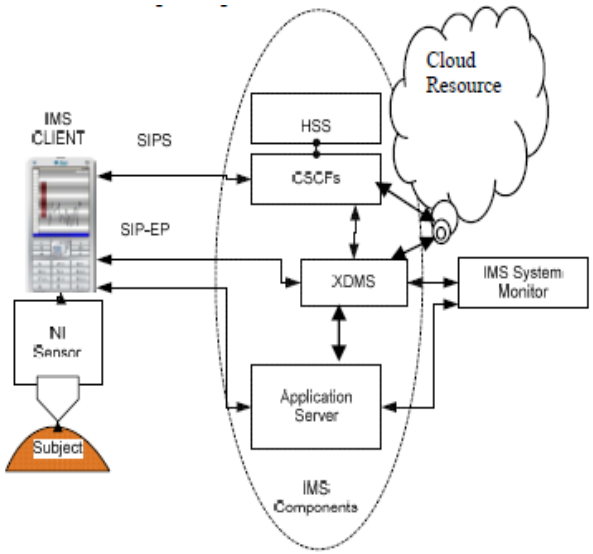
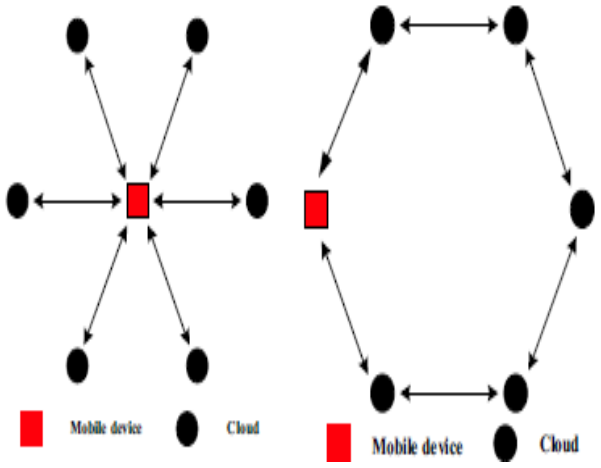


Figure-15: Model for IMS-Based Mobile Health Care Monitoring with Cloud Support

In this the NGN-IMS architecture has been exploited and the IP multimedia subsystem (IMS) mobile client is used. For the health monitoring a non-invasive (NI) sensor is used whose information like ECG, pulse rate, blood pressure etc. is gathered by a digital signal processor embedded into the IMS-client and for the execution of the appropriate algorithm on the multimedia digital data the cloud services has been exploited.
The approaches that could be used for offloading in mobile health care applications are self-reliant multi-cloud offloading system and multi-cloud offloading system [**33**]. It considers the existence of more than one cloud to offload the tasks for execution. In the self-reliant multi-cloud offloading system, the codes could be offloaded to remote servers one-by-one as shown in figure-16.



(a) Self Reliant Multi-Cloud (b) Multi-cloud
Offloading System Offloading System

Figure-16: Topology of Multi-Cloud Offloading System

In the multi-cloud offloading the concept of partial offloading is used. In this the clouds to which programs are offloaded could communicate among each other as well as with the mobile device for the overall execution of the code in the most efficient manner as shown in figure-16(b). But in the method explained earlier that is self-reliant multi-cloud system it was not possible for the cloud to communicate with each other, the only type of communication possible was between cloud and mobile device. Thus, with the support of the MCC the health care and health monitoring could be done and required measures could be taken to cure the diseases or to provide the emergency services.

I. Secured Offloading Design

The process of offloading from mobile smartphones to the remote server is prone to the security threats. Now-a-days it is a major challenge to perform the task of

offloading in a secured manner. A framework has been proposed for this purpose which talks about the elastic application that could have more than one weblets whose transfer is possible between smart devices and the remote server [**22**, **36**].
The model of elastic frame is explained in this as shown in figure-17. The main component of the device is device elasticity manager (DEM) that configures the application at initial start-up stage and at runtime. The major component of the cloud is cloud elasticity service (CES), it comprises of application manager, cloud manager, cloud sensing and cloud fabric interface (CFI) that avails web services for mobile devices. Now, the model provides security to the process
of offloading on both sides that are mobile device and cloud on the basis of the factors like authentication to provide secured environment of communication, secured migration of weblets and to authorize the weblets of cloud to use external web services.

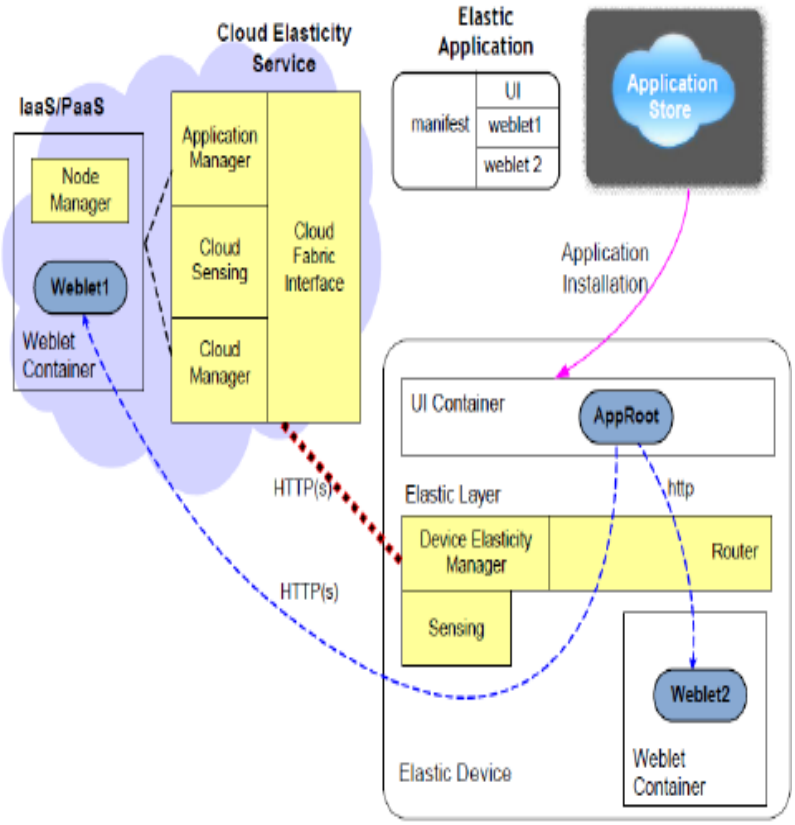


Figure-17: Architecture of Elastic Framework

For this purpose there is the need to develop security manager between the weblets. At the runtime the weblets could be migrated between the mobile device and the cloud, which is done in the fashion explained in figure- 18.

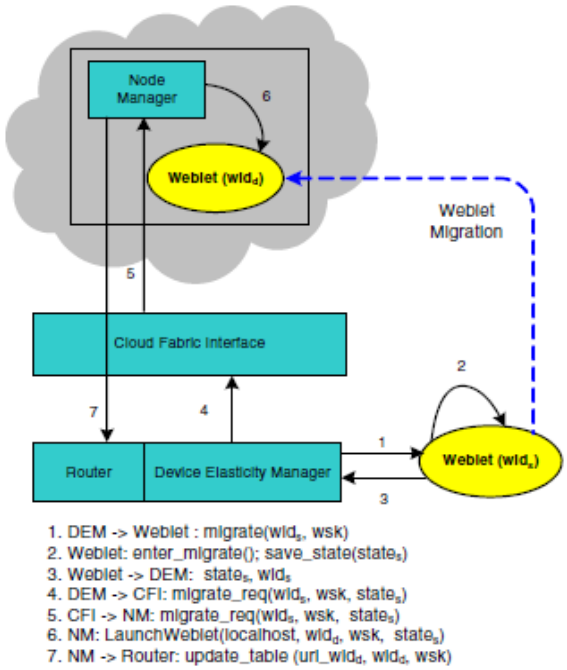


Figure 18: Migration of Weblet in Secured Fashion from Mobile Device to Cloud

The cloud weblets are also required to use external web services during the runtime of certain applications for which authorization is required to conduct the whole procedure in a secured fashion. This figure- 19 shows the methodology for realizing it.

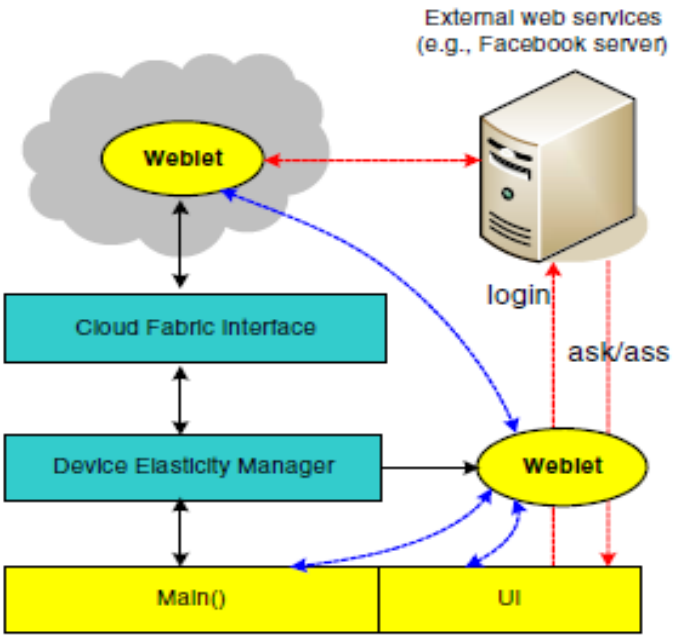


Figure-19: Process of Authorization of Weblets to Access External Web Services

The proposed model discussed the security threats while offloading and migrating the weblets and also provided the model to execute the whole procedure in a secured fashion and gives a trustworthy environment. Another model has been proposed for secured offloading that considers the approach of code partitioning on the basis of vulnerability [**38**]. It deals with the problem of security by offloading only that part of the application that is less vulnerable to the security threats and the vulnerable part is executed locally by the mobile device. The architecture of the approach is as shown in figure-20.

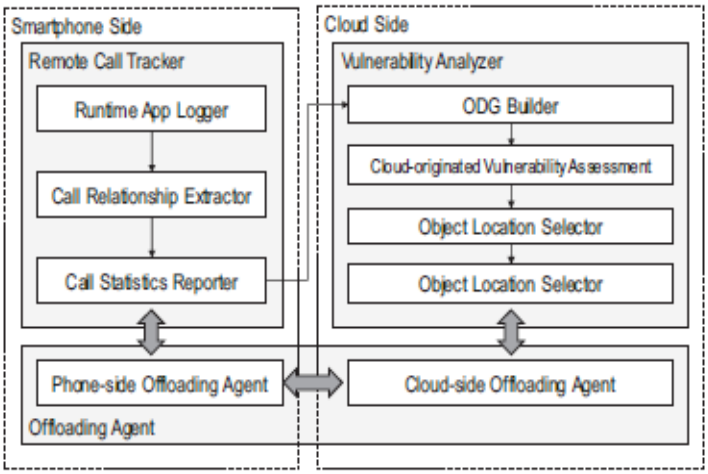


Figure 20: Architecture of Secured Offloading on the Basis of Vulnerability

Here, the first module that is remote call tracker partitions the application appropriately for further processing. The vulnerability analyser that is the second module analyses the vulnerability of the objects and the final decision has been taken by the third module that is offloading agent on the basis of vulnerability that whether it should be offloaded or not.

In this approach, object dependency graph (ODG) is prepared whose vertices signify the objects in an app and its edges signify their call relationships. All the minimum paths have been analysed based on depth first search (DFS) and all minimum cut-sets have been found and it has been inferred that if the edges of the minimum path of the graph is compromised irrespective if the status of other edges then it results in compromised target and if all minimum cut-sets have failed then it results in security threat. Hence, the graph gives the vulnerability of the
path between the objects of the apps and with this the final decision could be made that highly vulnerable objects should be offloaded locally so as to make it safe from at least cloud-originated vulnerabilities and other objects whose vulnerability is not that delicate could be offloaded to the remote server.

IV. ISSUES AND FUTURE RESEARCH DIRECTIONS

The MCC has been emerged as a great technology for mobile phone users. Its efficiency and user friendly approach makes it popular among common people. But still there is a huge scope of improvement in this area. Many issues are yet remain unexplored and that are open for the future researches. In this section we will address such issues and possible areas for the future research.

A. Inter-operability

There are multiple devices which are not communicated with the similar ones. The offloading can be done between the different systems with different capabilities. The interaction between the different systems should be hidden from the users [38].

B. Mobility and Fault Tolerance

The offloading is done between the server and the user device. The fault tolerance provides the continuous execution of application when there is any network issue or failure [39].

C. Privacy and Security

The privacy and security is considered as important issues because the data, which is offloaded from the mobile device to the cloud, can be accessed by the third party. The cloud is not in user's control. The encryption of data in an efficient way is still a research topic [40].

D. Cloudlet-Based Models:

In the field of MCC, the offloading is usually done from local server to the remote server that is cloud. But the mobile devices could roam anywhere, due to which the access of cloud from the local server becomes difficult as it is far away from it, which increases connectivity problems, execution time, communication cost etc. But, recent researches tried to make use of an intermediate layer in the form of cloudlet which is formed when there is no nearby cloud and used to keep the cloud near the mobile device and works as data centre for the cloud. But the question still remains about the situations in which it is deployed, offloading situations to the remote server that is far away etc.

E. Sensors Deployment:

The MCC have many applications like health care monitoring, intelligent calendar, forensic applications etc. But, the efficient cloud support depends on the data procured by sensors. But not much sensors are actually deployed for the same. So, the efficient deployment of the sensors is to be analysed that whether it could be done in random fashion or schematic fashion etc.

F. Secured Architectures of Offloading:

The security is a censorious challenge in offloading. The thread that is offloaded could be vulnerable to external threats. Hence, the selection of the thread to be offloaded for the secured offloading is still a hurdle to overcome.

G. Virtual Cloud Services:

The Bluetooth, zig-bee, Wi-Fi networks could be developed in the short range environment as a mode of communication among the devices residing in that range. The powerful mobile system that has large storage capacity and that could provide software services can be made to work as a virtual cloud for the mobile devices in the specified range.

V. CONCLUSION

The MCC has been emerged as a most promising solution for increasing the performance and efficiency of mobile devices as the applications has been offloaded to the remote server for the execution, this reduces the complexity and run time if it's done locally. This also proves out to be the solution for limited storage capabilities of the smart devices. The offloading techniques have been analysed in this paper along with its benefits, applications etc. In this paper the survey has been done on the existing models for offloading in MCC for various scenarios. And, finally some of the issues and future research directions in the field of offloading have been enlisted here.

REFERENCES

[1]. Choi, Min, Jonghyuk Park, and Young-Sik Jeong. "Mobile cloud computing framework for a pervasive and ubiquitous environment." *The Journal of Supercomputing* 64, no. 2 (2013): 331-356.

[2]. Barbera, Marco V., et al. "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing." *INFOCOM, 2013 Proceedings IEEE*, 2013.

[3]. Chun, Byung-Gon, et al. "Clonecloud: elastic Execution between mobile device and cloud." *Proceedings of the sixth conference on Computer systems*. ACM, 2011.

[4]. Chun, Byung-Gon, and Petros Maniatis. "Dynamically partitioning applications between weak devices and clouds." *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010.

[5]. Cuervo, Eduardo, et al. "MAUI: making smartphones last longer with code offload." *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010.

[6]. Dinh, Hoang T., et al. "A survey of mobile cloud computing: architecture, applications, and approaches." *Wireless communications and mobile computing* (2011).

[7]. Fekete, Krisztian, et al. "Energy-efficient computation offloading model for mobile phone environment." *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*. IEEE, 2012.

[8]. Fernando, Niroshinie, Seng Wai Loke, and Wenny Rahayu. "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing." *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. IEEE, 2011

[9]. Ferzli, Rony, and Ibrahim Khalife. "Mobile cloud computing educational tool for image/video processing algorithms." *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*. IEEE, 2011.

[10]. Flores, Huber, and Satish Narayana Srirama. "Mobile cloud middleware." *Journal of Systems and Software* (2013).

[11]. Gao, Bo, et al. "From mobiles to clouds: developing energy-aware offloading strategies for workflows." *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE Computer Society, 2012.

[12]. Gerla, Mario. "Vehicular cloud computing." *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*. IEEE, 2012.

[13]. Hoang, Doan B., and Lingfeng Chen. "Mobile cloud for assistive healthcare (MoCAsH)." *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*. IEEE, 2010.

[14]. https://aws.amazon.com/what-is-cloud-computing. Accessed on 5th March, 2016.

[15]. http://www.ibm.com/cloud-computing. Accessed on 5th March, 2016

[16]. http://www.microsoft.com/en-in/server-cloud/cloudos. Accessed on 5th March, 2014

[17]. Hu, Guoqiang, Wee Peng Tay, and Yonggang Wen. "Cloud robotics: architecture, challenges and applications." *Network, IEEE* 26.3 (2012): 21-28.

[18]. Jiao, Lei, et al. "Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities." *Future Network and Mobile Summit (FutureNetworkSummit), 2013*. IEEE, 2013.

[19]. Kemp, Roelof, et al. "Cuckoo: a computation offloading framework for smartphones." *Mobile Computing, Applications, and Services*. Springer Berlin Heidelberg, 2012. 59-79.

[20]. Kim, Jennifer. "Design and evaluation of mobile applications with full and partial offloadings." *Advances in Grid and Pervasive Computing*. Springer Berlin Heidelberg, 2012. 172-182.

[21]. Kovachev, Dejan, Tian Yu, and Ralf Klamma. "Adaptive computation offloading from mobile devices into the cloud." *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*. IEEE, 2012.

[22]. Kovachev, Dejan, Yiwei Cao, and Ralf Klamma. "Mobile cloud computing: a comparison of application models." *arXiv preprint arXiv:1107.4940* (2011).

[23]. Kumar, Karthik, and Yung-Hsiang Lu. "Cloud computing for mobile users: Can offloading computation save energy?" *Computer* 43.4 (2010): 51- 56.

[24]. Lagerspetz, Eemil, and Sasu Tarkoma. "Mobile search and the cloud: The benefits of offloading." *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 2011.

[25]. Lee, Jooyoung. "Pervasive forensic analysis based on mobile cloud computing." *Multimedia Information Networking and Security (MINES), 2011 Third International Conference on*. IEEE, 2011.

[26]. Ma, Xiao, et al. "Energy optimizations for mobile terminals via computation offloading." *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*. IEEE, 2012.

[27]. Nkosi, M. T., and F. Mekuria. "Cloud computing for enhanced mobile health applications." *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE,
2010.

[28]. Pocatilu, Paul, Felician Alecu, and Marius Vetrici. "Measuring the efficiency of cloud computing for elearning systems." *WSEAS transactions on computers* 9.1 (2010): 42-51.

[29]. Sanaei, Zohreh, et al. "Heterogeneity in mobile cloud computing: taxonomy and open challenges." (2013): 1- 24.

[30]. Srirama, Satish Narayana, Huber Flores, and Carlos Paniagua. "Zompopo: mobile calendar prediction based on human activities recognition using the accelerometer and cloud services." *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2011 5th International Conference on*. IEEE, 2011.

[31]. Su, Wei-Tsung, and Kiat Siong Ng. "Mobile cloud with smart offloading system." *Communications inChina (ICCC), 2013 IEEE/CIC International Conference on*. IEEE, 2013.

[32]. Wu, Huaming, Qiushi Wang, and Katinka Wolter. "Methods of cloud-path selection for offloading in mobile cloud computing systems." *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012.

[33]. Wu, Huaming, Qiushi Wang, and Katinka Wolter. "Mobile Healthcare Systems with Multi-cloud Offloading." *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*. Vol. 2. IEEE, 2013.

[34]. Yang, Seungjun, et al. "Fast dynamic execution offloading for efficient mobile cloud computing." *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013

[35]. Ye, Yunqi, et al. "A framework for QoS and power management in a service cloud environment with mobile devices." *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. IEEE, 2010.

[36]. Zhang, Xinwen, et al. "Securing elastic applications on mobile devices for cloud computing." *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009.

[37]. Zhang, Yuan, et al. "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing." *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*. IEEE, 2012.

[38]. Zhu, He, Changcheng Huang, and James Yan. "Vulnerability evaluation for securely offloading mobile apps in the cloud." *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*. IEEE, 2013.

[39]. Chun BG, Maniatis P (2009) Augmented smartphone applications through clone cloud execution. In: Conference on hot topics in operating systems, USENIX Association, pp 8–12

[40]. Kumar, Karthik, et al. "A survey of computation offloading for mobile systems."*Mobile Networks and Applications* 18.1 (2013): 129-140.

[41]. Shiraz, Muhammad, and Abdullah Gani. "A lightweight active service migration framework for computational offloading in mobile cloud computing." *The Journal of Supercomputing* 68.2 (2014): 978-995.

[42]. Byun, HwiRim, Boo-Kwang Park, and Young-Sik Jeong. "Mobile Agent Oriented Service for Offloading on Mobile Cloud Computing." In *International Conference on Computer Science and its Applications*, pp. 920-925. Springer Singapore, 2016.

[43]. Terefe, Mati B., Heezin Lee, Nojung Heo, Geoffrey C. Fox, and Sangyoon Oh. "Energy-efficient multisite offloading policy using Markov decision process for mobile cloud computing." *Pervasive and Mobile Computing* 27 (2016): 75-89.

[44]. Khan, Minhaj Ahmad. "A survey of computation offloading strategies for performance improvement of applications running on mobile devices." *Journal of Network and Computer Applications* 56 (2015): 28-40.

[45]. Folino, Gianluigi, and Francesco Sergio Pisani. "Automatic offloading of mobile applications into the cloud by means of genetic programming." *Applied Soft Computing* 25 (2014): 253-265.

[46]. Roy, Deepsubhra Guha, Debashis De, Anwesha Mukherjee, and Rajkumar Buyya. "Application-aware cloudlet selection for computation offloading in multi-cloudlet environment." *The Journal of Supercomputing* (2016): 1-19.