Mobility Aware Task Allocation for Mobile Cloud Computing

Bidoura Ahmad Hridita
MS Student
Institute of Information Technology, University of Dhaka

Mohammad Irfan
MS Student
Institute of Information Technology, University of Dhaka

Md. Shariful Islam
Associate Professor
Institute of Information Technology, University of Dhaka

ABSTRACT

The Mobile Cloud Computing is a promising technology that has provided a way to overcome the limitations of the mobile devices. The advancement of mobile devices technology has made the applications of these devices more complex and resource famished. Mobile cloud computing has created opportunities to execute these applications on the mobile devices by migrating the compute intensive task to the cloud. This migration of task to the cloud is not an easy task. The connectivity of the devices and the cloud is affected by the network inconsistency of wireless network. The servers on the cloud are heterogeneous in nature. Furthermore, the users are most of the time in mobile state which results in frequent change in association to access points. All of these make the selection of an optimal server to offload the task in cloud into a challenging work. In this paper, a comparative survey is provided for allocating task on the cloud along with their limitation. A mobility aware task allocation system for mobile cloud computing is also proposed. An optimization problem is formulated considering the workload and service rate of servers, network inconsistency, time to execute the task, mobility of the users etc. The proposed system aims to allocate task to the server where minimum response time is achieved in order to enhance users' quality of experience.

General Terms

Cloud computing

Keywords

Mobile cloud computing, Cloudlet, Task Allocation, Code offloading

1. INTRODUCTION

The number of mobile phone users in 2014 was 7100 million which will become 9.2 billion by 2020 as per the mobility report of November, 2015 from Ericsson [1]. Among them, the smartphone users will be 6.1 billion. This rapid growth of smartphone users is due to the increasing portability and capacity of mobile devices. The increase in mobile device users has created myriads of opportunities for mobile applications. The application developers are creating a large number of applications in various categories such as image processing, entertainment, social networking, health, business, real time monitoring etc. for mobile devices [2]. Though the mobile devices are capable of executing different advanced applications, these suffer from limitations like processing capabilities, battery lifetime, storage capacity etc. These limitations create a barrier over executing the resource famished applications mentioned earlier in the mobile devices. To unravel this problem, the researchers have introduced Mobile Cloud Computing (MCC). It refers to an infrastructure which amalgamates cloud computing, mobile computing and the wireless network. All the compute intensive tasks and the data storage are performed on the cloud resources to enable the execution of resource famished and greedy tasks on the mobile devices.

The process of migrating the compute intensive task to the remote cloud server for execution is called task or code offloading. As the real cloud provider may be located far away from the users, it can lead to an additional overhead like high latency and low bandwidth. For achieving high speed offloading purposes, trusted, resource rich computers called cloudlets are attached to the access points near the users of the mobile devices [3], [4].

The selection of the servers on the cloud is an immense research challenge considering the mobility of the users, heterogeneity of the cloudlets and the application request, network inconsistency etc. Existing offloading techniques such as MAUI, ThinkAir, CloneCloud etc. assume that the network performance is always consistent [5], [6], [7]. This is not true in real scenario. The network performance is affected by the mobility of the users. The performance gets degraded as the users move away from the access point. Some other techniques like MAPCloud, Location-aware task offloading etc. have considered the network inconsistency but did not predict the users' mobility to select the resources on the cloud [8], [9].

In this paper, at first a comparative survey is provided for allocating resource greedy task on the cloud from the mobile devices. After that a solution to the problem of where to offload the task from the available resources on the cloud from the mobile devices is proposed considering the users' mobility, network inconsistency and the heterogeneous nature of the cloudlets and the application request. The key contributions of this paper are as follows.

i. A comparative study is provided for the existing works addressing the issue of task allocation from mobile devices on cloud and the limitation of these works is presented.

ii. A two-tier architecture is proposed involving cloud, cloudlets and the mobile devices. The decision of selecting the server on the cloud will be taken on the cloud side to reduce the burden over the mobile devices.

iii. The solution considers users' mobility, network inconsistency and heterogeneity of the cloudlets.

iv. The mobility of the users is predicted based on the history of the movement of the users.

The paper is organized as follows. Section 2 briefly describes MCC architecture and applications, Section 3 present the comparative analysis of existing works related to this issue, Section 4 describes proposed solution for the problem and finally Section 5 concludes the paper with future plan.

2. MOBILE CLOUD COMPUTING

Mobile devices are gradually becoming an indispensable part of the human life with prodigious portability and flexibility. A chief deficiency of mobile computing is the resource scarcity. The Mobile Cloud Computing has overcome this deficiency. In this section, a brief description of MCC including the definition, applications and advantages of it is provided for better understanding.

2.1 Definition

The mobile cloud computing is a technology where mobile applications get the required large pool of resources from the cloud computing. The MCC forum defines the mobile cloud computing as "Mobile cloud computing as its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and MC to not just smartphone users but a much broader range of mobile subscribers" [18]. Satyanarayanan has described mobile computing as "information at fingertips anywhere, anytime" [19].

2.2 Architecture

Fig. 1 shows the basic architecture of MCC. Base stations and wireless access points connect the mobile devices to the mobile network. The connection and network interfaces between the mobile networks and devices are established and controlled through the base stations and access points. CPU connected to the mobile network receives the request from the mobile devices. Services like authentication, authorization etc. are provided by the mobile network based on the home agent and subscribers data from the database. Then the request from the subscriber is forwarded to the cloud through the Internet. The cloud controllers process the request from the subscribers and provide the requested cloud services to the mobile devices. The nearby cloudlets are used to increase users' experience since they may be distant from the clouds. The cloudlets are defined as "a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby mobile devices" [4]. The cloudlets act as the middleware between the mobile devices and the distant cloud. All the resource famished tasks are then offloaded to the cloudlets to achieve a low response time.

2.3 Applications

With the use of the mobile devices, the number of the mobile applications has grown enormously. The mobile applications that are using the cloud computing advantages are briefly discussed below.

- **m-learning** - m-learning is learning through the mobile devices. It is an amalgamation of the electronic learning and the mobility of the user. But it suffers from limitations like high cost of the mobile devices, limited educational resources, low transmission rate etc. These limitations are addressed with the help of cloud computing.

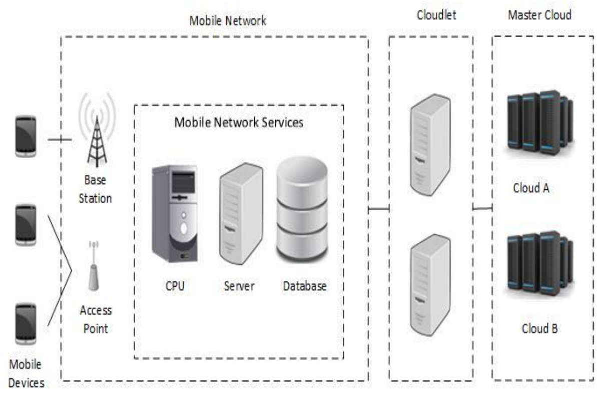

Figure 1: MCC architecture

- **m-commerce** - m-commerce helps to provide commerce with the help of mobile devices. It actually consists of the applications that require mobility such as mobile money transfer, mobile ticketing, mobile vouchers, coupons and loyalty etc. As the m-commerce includes mobility, it faces some problems like security, high complex configurations of mobile devices, low bandwidth. The integration of m-commerce with the cloud computing can help reducing these problems.

- **m-gaming** - The scope of the mobile games is generally small because of the limited processing and battery power of the devices. But the service provider can generate a large revenue from this market with the help of cloud computing. The mobile games can be completely offloaded to the cloud. This reduces the computation task on the mobile devices and saves the energy.

2.4 Advantages

MCC has many features that are advantageous for the end users and the cloud providers. These are listed below.

- **Processing power** - Mobile devices have limited resources. Naturally the heavy weighted computations are not feasible in these devices. The MCC provides the devices a humongous resource pool. The heavy weighted computations are performed on the cloud side. Thus the limited processing power of the devices does not impede the compute intensive application on the devices.

- **Battery power** - Another important shortcoming of the mobile devices is the limited battery power. The power consumption can be reduced by enhancing CPU performance, using disk and screen in an efficient manner. But these incur changes in the device structure with increased monetary cost. The task offloading to the mobile cloud helps address this issue. As the computation intensive tasks are offloaded to the cloud, long execution time on the mobile devices are avoided which in turn increase the battery life of the devices.

- **Data storage capacity** - Storage capacity is another constriction of mobile devices. Devices naturally have a short storage capacity. The cloud computing can be used to escalate the storage capacity of the devices by using the huge storage capacity of the cloud through the wireless network.

- **Reliability** - The cloud computing intensify the reliability of the mobile devices as the data are stored on several computers on the cloud. Ultimately the chance of losing the data is scaled down. In addition to this, the design of the MCC can be done in ways to make it a data security model for the users and the providers of the services.

3. STATE-OF-THE-ART TASK ALLOCATION MECHANISMS

As MCC is a nascent technology, many researchers have been showing interest in this field. Task offloading is a major issue in MCC to overcome the limitations of mobile devices. Different efforts have been taken by the researchers in recent years like [10], [11], [12], [5], [6], [7], [8], [9], [13], [14], [15] etc. Cuervo et al. [5] proposed a system called MAUI which determines the offloadable task of the application either statically or dynamically. This system aims to find the part of an application which is compute intensive and thus can be executed remotely on the cloud to save the energy of the mobile devices. Chun et al. [7] proposed CloneCloud which partitions the application between mobile devices and the clouds. It stores the execution time, energy consumption etc. of the task after executing it under different conditions such as network characteristics, CPU speeds etc. From these, a partition is picked at runtime and offloaded to the cloud. All these works focused on finding the compute intensive part of the application to migrate it to the remote cloud in order to reduce the burden over mobile devices.

The issue of selecting the servers from the cloud to execute the resource famished task is addressed by [8], [13], [9], [14], [15], [16], etc. Brief description of these works is narrated below with comparison among these.

3.1 MAPCloud

In MAPCloud [8], Rahimi et al. considered a two-tiered architecture consisting of local cloud and public cloud. They considered the application request as a workflow of task. By assuming the location was known beforehand, they selected the near optimal solution to allocate to the mobile devices from the cloud which satisfies multidimensional quality of service or QoS constraint like price, power and delay. But they did not consider mobility of the users to find the optimal resource on the cloud. The user may be far away from a resource in time of returning the result if a resource near to the user at a given time is selected as the optimal solution. Thus mobility is an important factor in case of choosing a resource on the cloud.

3.2 MuSIC

To address the issue of mobility, Rahimi et al. [13] expanded their work MAPCloud where they considered the application request as location-time workflow. The optimization algorithm was designed to find the resource from the cloud assuming the location and the time is known from the history while satisfying the constraints over multidimensional QoS like price, power and delay. Though they considered users' mobility in this work, they did not consider the velocity and change of direction of the user. Besides mobility, they have assumed the resources on the clouds are homogeneous in nature in both MAPCloud and MuSIC which is not in real life scenario.

3.3 ENDA

In ENDA [14], Li et al. selected the server from the cloud based on the user track prediction, server load and network quality. The user track is predicted from the history of mobility of the users stored in the database. The server that can fulfill the desired response time with lowest latency on the predicted route is selected to execute the task instead of the mobile device. Like MuSIC, this work also did not consider the velocity and the change of direction of users, and heterogeneity of the cloudlets.

3.4 Context Sensitive Offloading Scheme

Jhou et al. [15] considered the context of the mobile devices like network condition, location, workload etc. to offload the task to a server on the cloud as these change continuously with the movement of the mobile devices throughout the day. The server with lowest cost of execution on the cloud is selected as the location to execute the task remotely. Though they considered the network condition, workload of the clouds, they did not consider the mobility of the users, heterogeneity of the cloudlets etc. to select the server.

3.5 Mobi-Het

Asma [16] considered the velocity and direction change of the users, and heterogeneity of the cloudlets to optimally select the server on the cloud in her thesis work Mobi-Het. She predicted users' speed and direction using smooth random probability model. She formulated an optimization problem considering the time of execution on the cloud, workload, received signal strength etc. to offload the task to the server on the cloud.

3.6 Online Algorithms for Location-aware Task Offloading

All of the works mentioned earlier did not consider the limitation of user access on the wireless access point. Xia et al. [9] considered this factor to select the server on the cloud. First their system finds out which access points are free to associate with, then the server attached to the available access point with lowest energy cost is selected as the location to offload the task on the cloud.

3.7 Comparison

In this section a comparative study is provided among the works described earlier. To compare the works, considered parameters are users' mobility, execution location, load balancing, network inconsistency, fault tolerance, user access limitation on AP, cloudlets' heterogeneity, etc. The summary of this comparative study is presented in Table 1. The criteria of the table depict the following.

Execution location –where to offload the task?

Users' mobility – whether the mechanism predicts the mobility pattern of users?

Load balancing – either the mechanism considers the load constraint of the resources or not?

User access limitation on AP – does the mechanism consider the constraint of user access limitation of AP?

Cloudlets' heterogeneity – if the mechanism considers the cloudlets as homogeneous or heterogeneous

Fault tolerance – whether the mechanism has fault tolerant capability?

Network inconsistency – if the mechanism considers the network inconsistency for resource allocation?

Table 1. Comparison among the proposed solutions

	MAPCloud [8]	MuSIC [13]	ENDA [14]	Online Algorithms for Location-aware Task Offloading [9]	Context Sensitive Offloading Scheme [15]	Mobi-Het [16]
Execution Location	Master cloud/cloudlet					
Users' mobility	×	✓	✓	×	✓	✓
Load balancing	×	×	✓	×	×	✓
User access limitation on AP	×	×	×	✓	×	×
Cloudlets' heterogeneity	×	×	×	×	×	✓
Fault tolerance	×	×	✓	×	✓	×
Network inconsistency	×	×	✓	×	✓	✓

The task can be offloaded to master cloud or cloudlet. The cloudlets can response to the users faster than the master cloud as this is located far from the users most of the time. All of the works have considered master cloud and cloudlet as the execution location. Device cloud is considered in Context Sensitive Offloading Scheme [15] where the mobile devices form an ad-hoc cloud as per the requirement.

Users' mobility is an important issue which has a large impact on task allocation. The task should be allocated to resources that can satisfy users' quality of experience. If the task is allocated to a resource situated near to users at the time of offloading without considering their mobility, users' quality of experience may degrade as they may become distant to the resource for their mobility pattern at the time of returning the result. Therefore, users' mobility has to be predicted beforehand to take a better offloading decision. MuSIC [13], ENDA [14], Context Sensitive Offloading Scheme [15] and Mobi-Het [16] have considered mobility of users. Among these, the change of velocity and direction of users are considered only in Mobi-Het. In real scenario, no user moves along same direction with a constant velocity. To make a better optimal decision for offloading, these should be considered.

In case of offloading a task on the cloud, load balancing should be carefully handled. Offloading a task to an overloaded resource would result in increased response time. To minimize the response time, load of the resources should be balanced. Only ENDA [14] and Mobi-Het [16] have considered load balancing at the time of allocating task to a resource on the cloud.

All the works except Mobi-Het [16] have assumed the resources on the cloudlets have similar processing capacity which is not true for real scenario. The processing capability, storage power, etc. of the resources are different.

Online Algorithms for Location-aware Task Offloading [9] has considered users access limitation on access point. Access points can serve to a limited number of users at a time. A user can no longer associate to an access point if this number of users exceeds. Thus, a user has to wait in order to associate to an access point until an already associated user disassociates. This results to a longer response time degrading users' quality of experience.

4. PROPOSED TASK ALLOCATION SCHEME

In this section the system architecture of the proposed solution is introduced. Then, an optimization problem is formulated to select the server and a greedy algorithm of is provided for the proposed solution.

4.1 System architecture

Fig. 2 depicts the architecture of the proposed system. The system consists of two tiers of which tier 1 is master cloud and tier 2 is local cloud. Master cloud is the cloud with a large resource pool and high computation power. These resources are highly available at any time to the user. But as the users may remain far away from the master cloud due to their mobility, local cloud is considered as the second tier to ensure better service for the users. The local cloud consists of the cloudlets each of which is attached to a wireless access points and a middleware entity. These cloudlets are situated near the users with limited computation capability and resource pool compared to the master cloud. All of these cloudlets are heterogeneous in nature that means these have different computation power. The middleware entity consists of database profiler, request scheduler and optimizer. The functions of these are narrated below.

- **Database** - The database stores mobility history of the users, location of each cloudlet and range of these. The mobility history is kept as location with the duration of time spent in that location.
- **Profiler** - The profiler keeps record of the current workload and service rate of all the cloudlets and the servers in master cloud.
- **Request Scheduler** - The request scheduler receives application offload request from the mobile devices. It collects mobility information of the mobile users from database and servers from profilers. After getting this information, it consults the optimizer to solve the optimization problem.
- **Optimizer** – The optimizer solves the optimization problem proposed in this paper to find the best server on the cloud to offload the task.

Mobile devices in the proposed system have to convey very small amount of information to the cloud so that the battery life duration of the devices can be increased. The responsibility of making offloading decisions is shifted to the local clouds. In order to maintain the history of mobility of the users, devices need to send its location information to the cloud. The application request sent from a mobile device includes mobile ID, present location, number of instruction of the task and maximum allowable time to execute the task. This request is received by the request scheduler in middleware entity. It then consults the database to check the mobility history of the user. From this history, it selects the location where the user spends longest time. Then it finds the cloudlets situated at the selected location. The recent information of these cloudlets found near the location like current workload, service rate etc. is collected from the profiler. At last, the optimizer runs the optimization algorithm to find the optimal server to allocate the task. Then the request scheduler offloads the task to that server.

4.2 Problem Formulation

Main goal of the solution is to find a server that will return the result of the offloaded task with lowest amount of time. In other words, the response time R of the server is to be minimized because the QoS constraint enforces a prerequisite of executing the task within the maximum allowable time *t*.

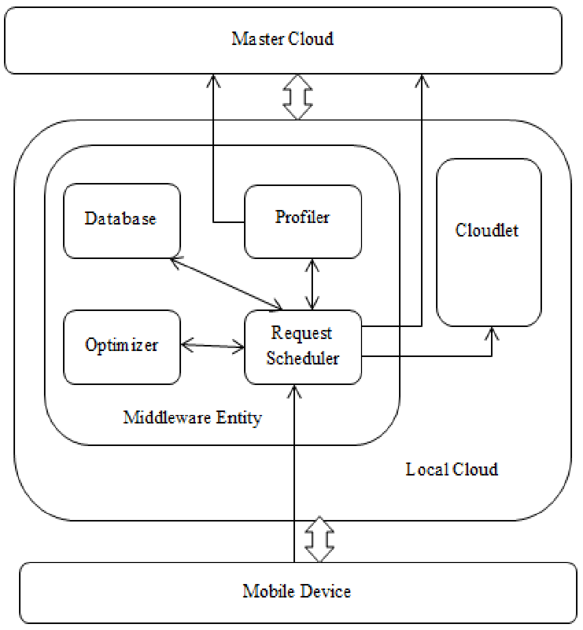

Figure 2: Architecture of proposed system

By assuming the workload *w*, service rate μ of cloudlets, number of instruction *l* and maximum allowable time *t* of execution of a task *T* is known beforehand, the response time *R* of a task *T* can be expressed as the time to send the task t^s, time to execute the task t^e and the time to receive the task t^r. Therefore, the response time *R* can be written as follows.

$$R = t^s + t^r + t^e \tag{1}$$

Table 2. Data Rate vs RSSI

Data Rate, d (Mbps)	RSSI (dBm)
1	-81
2	-79
5.5	-77
11	-75
6	-81
9	-80
12	-78
18	-76
24	-73
36	-69
48	-65
54	-64

The mac layer delay for transmitting the data to the cloudlets is considered. The mac layer delay will be high if the number of nodes associated to an access point is high as the defer time will increase for large number of users. Ultimately, the response time for a particular device will increase.

Besides this, the values of t^s and t^r is affected greatly by the received signal strength (RSSI) value. If the RSSI value of an access point is low, the data rate for the connection would be poor affecting the time to send the task and receive the result. The mapping table for RSSI values and the corresponding data rate is shown in Table 2 [17] which can be used to calculate the data sending and receiving time. Denoting the data to send as f^s and data to receive as f^r, for a task *T* and the achievable data rate as *d*, the t^s, t^r and t^e can be calculated from below.

$$t^s = f^s/ d + d\_mac \tag{2}$$

where,

$$d\_mac = d_{backoff} + d_{defer} \tag{3}$$
$$t^r = f^r/d \tag{4}$$
$$t^e = l/\mu \tag{5}$$

After receiving an application request, the request scheduler invokes the optimizer to carry out the following objective function for finding the optimal server to offload the task.

$$\min \textstyle\sum_{ij} R_{ij}x_{ij} \tag{6}$$

s.t.

$$\textstyle\sum_j w_{ij}x_{ij} \le w_i^{cap} \qquad \forall i \tag{7}$$

$$\textstyle\sum_i x_{ij} = 1 \qquad \forall j \tag{8}$$

$$\textstyle\sum_i R_{ij}x_{ij} \le t_j \qquad \forall j \tag{9}$$

Here, equation 6 is the objective function which is to be solved by the optimizer of middleware entity in the local cloud. It is assumed that the local cloud has enough computing resources to solve this problem within the specified time. Equation 7 is the constraint for workload distribution. The workload *w* of a cloudlet should not exceed its workload capacity w^{cap} after allocating the task to it. Equation 7 specifies this condition. The workload capacity of a cloudlet can be calculated through summarizing the product of size of the virtual machines in the cloudlet and their size as in [16]. If a cloudlet has *V* number of instances of virtual machines and the size is *S*, the workload capacity of a cloudlet can be calculated as follows.

$$w^{cap} = \textstyle\sum V_i \times S(i) \tag{10}$$

If the current number of instances of the virtual machine is $V^{current}$, the current workload of a cloudlet can be expressed as

$$w = \textstyle\sum V_i^{current} \times S(i) \tag{11}$$

The task can be offloaded to only one server. Equation 8 specifies this unique assignment of task to a server. Finally,

Table 3. Algorithm for Task Allocation

```
1:  input: mobileID, location, maxAllowableTime, serverInfo
2:  output: serverID
3:  procedure TASK ALLOCATION
4:      location ← GetLocationFromDB(mobileID)
5:      points[n]← GetPoints(Location)
6:      initialize trajectory ← null
7:      initialize max ← -1
8:      for i ← 1 to n do
9:          time ← GetSpentTime(point[i])
10:         if max < time then
11:             trajectory ← point[i]
12:         end if
13:     end for
14:     servers[m] ← GetServers(point[i])
15:     min ← maxAllowableTime
16:     serverID ← -1
17:     for i ← 1 to m do
18:         resposeTime ← GetServerResponseTime(serverInfo)
19:         if min > responseTime then
20:             serverID = server[i]
21:         end if
22:     end for
23:     return serverID
24: end procedure
```

equation 9 specifies the constraint for QoS that is the task should be completed within the maximum allowable time to execute the task.

This problem is an NP-Hard problem. Therefore, a greedy algorithm is proposed to solve this problem. In the following section this algorithm is presented.

4.3 Greedy Solution

The algorithm for the proposed solution is shown in Table 3. The algorithm starts by finding the mobility history of the users. From this history, the location where the users spend most of their time during mobility is selected. Servers situated at that location is retrieved from the database in line 5. Following this, algorithm enters a loop which calculates the response time for the servers retrieved from the database. The response time is calculated through the function GetResponseTime having parameters current workload, maximum workload and service rate of the server, time to send the task, time to receive the result. If the calculated response time is greater than the maximum allowable time of the task, the loop will continue. Otherwise, the loop is terminated and the server is returned as the solution.

5. CONCLUSION AND FUTURE WORK

In this paper, a system to offload a task from mobile devices to cloud in a two-tiered environment is proposed to increase the computation capacity of the devices. An optimization problem for this is formulated. As this work is still ongoing, the experimental result could not be provided in this paper. Future plan is to implement the solution and conduct a wide-ranging experiment to evaluate the performance of the proposed system.

6. ACKNOWLDGEMENTS

The authors would like to thank the honorable reviewers for reviewing this paper and their valuable and constructive comments in case of improving this paper.

7. REFERENCES

[1] Ericsson Mobility Report, Nov 2015, http://hugin.info/1061/R/1872291/659558.pdf

[2] N. Fernando, SW. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol. 29, pp. 84–106, Jan 2013.

[3] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudelets: bringing the cloud to the mobile user," in *Proc. of third ACM Workshop on Mobile cloud computing and services, 2012.* MCS 2012, pp. 29-36.

[4] M. Satyanarayanan, P Bahl, R Caceres, and N Davies, "The Case for VM-Based Cloudlets in Mobile Computing," in *IEEE Pervasive Computing,* vol. 8, pp. 14-23, Oct 2009.

[5] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of International Conference on Mobile Systems, Applications, and Services, 2010. MobiSys 2010.* pp. 49–62.

[6] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. of IEEE INFOCOM*, 2012, pp. 945–953.

[7] B. Chun, M. Naik, S. Ihm, A. Patti, and P. Maniatis, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. of European Conference on Computer Systems*, *2011*. EuroSys 2011, pp. 181–194.

[8] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, 2012, pp. 83-90.

[9] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online Algorithms for Location-Aware Task Offloading in Two-Tiered Mobile Cloud Environments," in *Proc. of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 109-116.

[10] S. Ou, K. Yang, and J. Zhang, "An effective offloading middleware for pervasive services on mobile devices," in *Journal of Pervasive and Mobile Computing*, 2007, vol. 3, pp. 362-385, August 2007.

[11] M. V. Barbera, S. Costa, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proc. of IEEE INFOCOM*, 2013, pp. 1285-1293.

[12] E. Gelenbe, R. Lent, and M. Douratsos, "Choosing a local or remote cloud," in *Second Symposium on Network Cloud Computing and Applications*, 2012, pp. 25-30.

[13] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *IEEE Sixth International Conference on cloud computing (CLOUD)*, 2013, pp. 75- 82.

[14] J. Li, K. Bu, and B. Xiao, "ENDA: embracing network inconsistency for dynamic application offloading in mobile cloud computing," in *Proc. of the second ACM*

SIGCOMM workshop on Mobile Cloud Computing, 2013, pp. 39-44.

[15] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service," in *Proc. of the IEEE 8th International Conference on Cloud Computing (CLOUD)*, 2015, pp. 869-876.

[16] Asma Enayet, "Mobility Aware Optimal Resource Allocation in Heterogeneous Mobile Cloud Computing," MS thesis, Department of Computer Science and Engineering, University of Dhaka, 2015.

[17] What is the relationship between data rate, SNR and RSSI, http://community.arubanetworks.com/t5/Controller-Based-WLANs/What-is-the-relationship-between-data-rate-SNR-and-RSSI/ta-p/178312

[18] Mobile Cloud Computing Forum, http://www.mobilecloudcomputingforum.com

[19] M. Satyanarayanan, "Mobile computing: the next decade," ACM SIGMOBILE Mobile Computing and Communications, vol. 15, issue 2, April 2011, pp. 2-10.