Saving Energy in Mobile Devices Using Mobile Device Cloudlet in Mobile Edge Computing for 5G

Tshiamo Sigwele, Prashant Pillai, and Yim-Fun Hu,

***Abstract*—In the future, the next generation cellular networks like fifth generation (5G) will comprise of billions of devices with various applications running on the devices. These applications are computer intensive and drain a lot of battery when executed in the mobile device itself. Mobile Edge Computing (MEC) has been proposed to solve these problems by offloading computation tasks of an application to the edge server in the radio access network (RAN). The conventional MEC framework suffer greater delays which is not suitable for 5G. In this paper, a new MEC framework in heterogeneous networks (HetNet) called MECH is proposed where a mobile device with limited resources has an option of offloading some of its tasks to a group of nearby mobile devices while considering the transmission power, quality of service (QoS) and state of charge (SoC) of the mobile battery. The simulation results demonstrates that the proposed framework extend battery life and reduces delays compared to the traditional MEC paradigm.**

***Index Terms*—Cloud Computing, Mobile Edge Computing, Energy-efficiency, 5G, Computation Offloading.**

1. Introduction

In 2020, billions of devices like smart phones, tablets and Internet of Things (IoT) will be connected to the fifth generation (5G) cellular networks. These devices will comprise of a plethora of applications like mobile gaming, health monitoring and augmented reality. These applications are computer intensive and drains a lot of battery in the mobile devices. The mobile devices can hardly cope due to limitations in terms of battery life, storage, memory and processing power [1]. Extended battery life is one of the key requirements by mobile phone users as compared to memory, storage and display size, as such there is a need for improvement of energy efficiency (EE) in mobile devices. One possible approach is to offload the application computation to the remote public clouds such as Amazon EC2 and Windows Azure using mobile cloud computing (MCC) which will save some amount of energy in the mobile device. These cloud centers provide virtually unlimited computation capacity to augment the processors in mobile devices. However, the communication between mobile users and remote cloud centers is often over a long distance, adding to the latency in cloud computation. To overcome this limitation, mobile edge computing (MEC) [2], also termed FOG computing [3] was proposed as shown in Fig. 1. In the MEC framework, cloud computing capabilities are provided within the radio access network (RAN) in close proximity to the mobile devices. In other words, with the aid of MEC, mobile devices are enabled to offload their application tasks to the MEC servers on the edge of the network, rather than utilizing the servers in the core network in the cloud datacenters. This MEC paradigm can provide low latency, high bandwidth, high computing agility and improve the energy performance of the mobile devices [2].

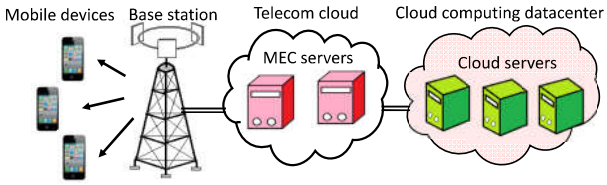

Figure 1. An illustration of MEC architecture.

The drawbacks of the current MEC system in offloading mobile device tasks to the MEC server is that there are still delays incurred between the mobile device and the MEC server caused by data transmission. Also, more energy is consumed in the mobile device during transmission of tasks to the MEC server. On the other hand, heterogeneous radio access networks (HetNets) have been recently proposed as a paradigm shift for 5G networks and a promising solution for increasing capacity in cellular networks. In HetNets, macro base stations (BSs) are overlaid by small cell BSs (micro, pico, relay, femto) where the macro BS provides coverage while the small cells are to increase capacity in hotspot areas. In this paper, MEC is incorporated within HetNets (Herein termed MECH) to minimized the energy consumption in the mobile device. In the proposed MECH framework, the mobile device with limited resources has the option of offloading its tasks to nearby mobile devices called a cloudlet [2]. The contributions of this paper are as follows:

i) An energy efficient MEC offloading scheme in HetNet called MECH is proposed to save energy in the mobile device. The mobile application is broken into modules/tasks by the developer which are then offloaded to the nearby mobile devices not far away in the MEC server. The transmission cost and delay are taken into consideration to improve quality of service

(QoS). As such, energy consumption and delays will be reduced.

ii) An architecture for MECH is proposed where the mobile devices also act as servers for processing tasks from the client mobile device with limited resources.

iii) A rigorous simulation study is conducted for validating the proposed scheme, which shows a significant performance improvement.

The rest of the paper is organised as follows: Section II discusses the related works in EE in mobile devices using MEC paradigm. Section III present the proposed MECH framework and problem formulation. The proposed architecture for MECH and the offloading scheme are presented in this section. The simulation results and discussion are provided in Section IV. Finally, the concluding remarks are presented in Section V.

2. Related Work

The reader is directed to a comprehensive survey on MEC in [4] [5]. The related work is divided into two categories, EE using MEC server as the resource provider and also mobile devices as resource providers. .

2.1. Offloading to MEC Server

The author in [6] investigates a green MEC system and develop an effective computation offloading strategy. The execution cost, which addresses both the execution latency and task failure, is adopted as the performance metric. Nevertheless, the author assumes that the battery capacity is sufficiently large which is impractical, also the author ignores the execution delay caused by the MEC server. Chen L. *et al.* in [7] addresses the challenge of incorporating MEC into dense cellular networks, and propose an efficient online algorithm, called ENGINE (ENergy constrained offloadINg and slEeping) which makes joint computation offloading in order to keep the energy consumption low. However, the author assumes that traffic among BSs is equally distributed while traffic is randomly distributed in reality as with our proposed framework. Zhang K. *et al.* in [2] proposed an energy efficient computation offloading (EECO) mechanisms to minimize the energy consumption of the offloading system, where the energy cost of both task computing and file transmission are taken into consideration. However, the author does not show the impact on the response time of an application offloaded to the MEC server which is clearly shown in this paper. The author in [8] proposed a distributed computation offloading decision making problem in MEC among mobile device users as a multi-user computation offloading game. However, the application to be offloaded is assumed to be atomic (the application can not be divided into modules) as such, the whole application is either executed locally or the whole code is send to the MEC server which can incur more transmission costs whereas in our proposed framework, an application can be divided into smaller tasks called classes. For all the above offloading schemes, offloading application tasks to the MEC server will incur delays and increase energy consumption during transmission.

2.2. Offloading to Mobile Devices

One way of solving the delays of offloading tasks to the MEC is to offload tasks to the nearby mobile devices acting as resource providers. G. Huerta-Canepa *et al.* in [9] proposed a framework to create Ad Hoc cloud computing providers. This framework takes advantage of the pervasiveness of mobile devices, creating a cloud among the devices in the vicinity, allowing them to execute jobs between the devices. However, the author assumes that the saving in processing time implies a saving in energy which have not been validated and this is being validated in our proposed scheme. The authors in [10] proposed a framework called DroidCloudlet as a new cloudlet architecture using commodity mobile devices, which can cooperate to support resource-poor devices with abundant resources in the other devices. DroidCloudlet enables application developer to define resource-hungry classes that might need to be offloaded. The authors in [11] proposed CANDIS, a framework that can distribute computing tasks to a computing cloud consisting of mobile devices running Android. It can partition tasks based on the computation power of the participating clients. However, CANDIS only cater for Android mobile devices while our proposed scheme cater for all the devices. In Hyrax [12], Marinelli *et al.* developed a platform derived from Hadoop that supports offloading tasks to Android smartphones and stationary servers. By scaling with the number of devices and tolerating node departure, Hyrax allows applications to use distributed resources abstractly; regardless of the physical nature of those devices. Nevertheless, Hyrax only applies to Android devices while our proposed scheme applies to all devices. Busching *et al.* in [13] proposed DroidCluster framework which implemented a small feasibility study using Android phones that are connected using either Wi-Fi or virtual Ethernet (via USB) and showed that Android systems are PC-like enough so that it is easily possible to deploy standard tools and mechanisms from the stationary computing world to successfully distribute computational tasks among them. All the above schemes does not involve parallel processing of tasks while in the proposed scheme, there is parallel processing of tasks within other mobile devices.

3. Proposed MECH Framework

3.1. System Architecture

The proposed system architecture is shown in Fig. 2. Consider a set of mobile users denoted as $\mathcal{K} = \{k : k = 1, 2, ..., N_{UE}\}$. We consider that each mobile device user runs an application which can be split into several tasks by the developer. Each task T_k of user device k can be

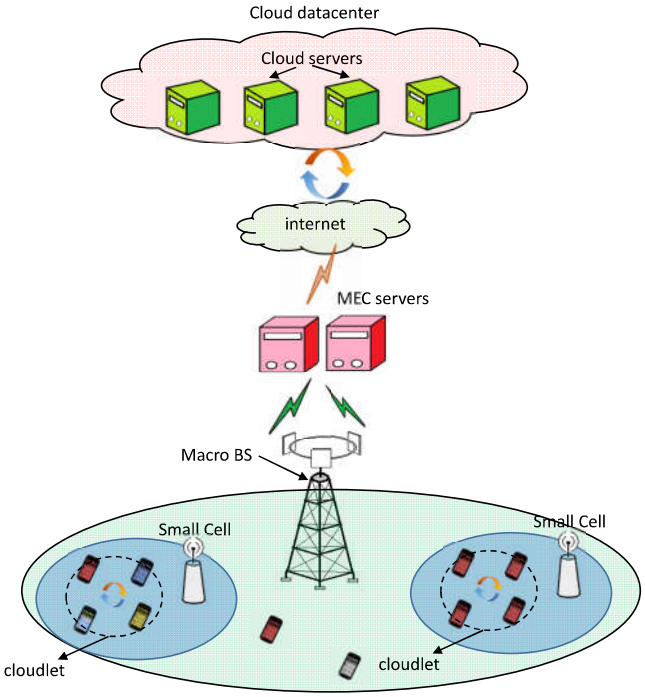

Figure 2. Proposed MECH architecture.

executed either locally on the mobile device or remotely on nearby recipient mobile devices which forms a cloudlet. The mobile device k has many interfaces and can connect to many devices simultaneously. Denote $a_{k,m} = \{0, 1\}$ as the offloading decision profile of user k. When $a_{k,m} = 1$ it means device k uses mode m to execute task T_k or $a_{k,m} = 0$, meaning there in no execution. The variable $m = \{1, 2\}$ is the user device chosen modes which are computing locally when $m = 1$ or offloading to the mobile device cloudlet, otherwise.

3.2. MECH Offloading Framework

This section will describe the MECH offloading framework in detail. The framework is shown in Fig. 3. The framework comprise of the client mobile device which have tasks to be offloaded and recipient devices forming a cloudlet where the tasks of the client device will be executed. The tasks from the client mobile device are executed in parallel fashion in the recepient mobile device cloudlet. The system components are as follows:

i) **Mobile application:** This is an elastic application (divided into tasks called classes by the developer and the classes are offloadadble).

ii) **Device profiler:** Collects mobile device hardware context at runtime and pass the information to the offloading agent for decision making of whether to execute locally or offload. The profile context include the battery state of charge (SoC), average central processing unit (CPU) utilization and memory usage.

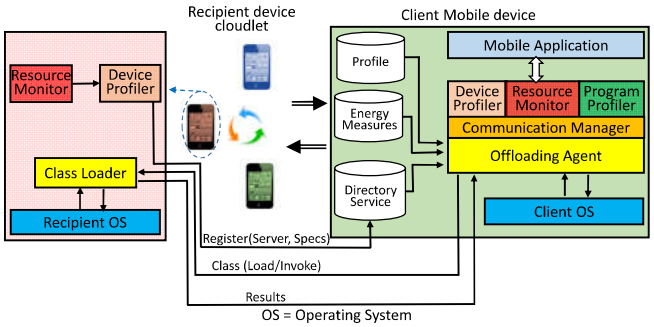

Figure 3. Proposed MECH framework.

The device profiler in the recipient devices collects the hardware context (server name, server computation speed, server SoC sate) of the remote device and register them in the directory services in the client device.

iii) **Resource monitor:** Resides in both the recipient and the client devices. It collects network related context at runtime and pass the information to the decision engine. The context include network connection state, bandwidth, and signal strength.

iv) **Program profiler:** The program profiler tracks the execution of the program and collects program context information such as total instructions executed, execution time, memory allocated. The profile is updated at every invocation and it is stored in the mobile device database.

v) **Communication manager:** It creates and maintains connection between the client and the server side. It serializes the code on the client side and deserializes the request from the client at the cloud side. It also keep the client and the server in sync.

vi) **Offloading agent:** This is the main module of the MECH framework. The Offloading agent consists of a set of cost estimation models like the delay or execution time and the energy models. Based on the received context information and the energy model, the offloading manager decides on when, where and how to offload the task/class. When a new task arrives, the offloading agent intercept it before it can be executed in the client operating system (OS). A check is then made to see if the task can be offloaded or executed locally. If there are devices in the cloudlet (a check is made to see if there are devices registered in the directory services) which are registered in the directory service with enough resources to execute the task, the offloadable class codes is then uploaded to the class loader of the recipient device which then execute the class code in the recipient OS, and after execution, the results are loaded back to the offloading agent in the client device.

vii) **Class loader:** This is located in all recipient devices. It receives the class codes from the offloading agent and then execute them in the recipient OS and send back

the results to the offloading agent.

viii) **Profile:** Stores the device context information including the battery SoC, average CPU utilization, memory usage and computation speed.

ix) **Energy measures:** Contains the energy cost models calculated in (2) and in (8) required for the offloading decision.

x) **Directory service:** This stores a list of active recipient mobile devices with their specifications like name, SoC level and memory status. A device in close proximity to the client device will advertise itself as a server and will be stored in the directory service.

3.3. Computation Model

Each task of mobile user k is denoted as $T_k = (B_k, D_k, t_k^{max})$. Here B_k denotes the size of computation input data in bytes (e.g., the program codes and input parameters) involved in the computation task T_k and D_k denotes the processing requirement in million instructions required to accomplish the computation task T_k. The variable t_k^{max} denotes the maximum latency required by the computation task T_k or the execution deadline in milliseconds (ms).

3.3.1. Local Computation. Local computation is when the mobile device k executes its computation task T_k locally on the mobile device. Denote F_k^l as the computation capability of the mobile device user k in million instructions per second (MIPS). It is assumed that mobile devices can have various computation capability. The execution time for executing task T_k for user k can be expressed as

$$t_k^{local} = \frac{D_k}{F_k^l} \tag{1}$$

The energy expended by the mobile device user k for local computation can be expressed as

$$e_k^l = t_k^{local} P_a \tag{2}$$

where P_a is the power consumed by the device when active. It is assumed that the only one application is running in the mobile device and other application are deactivated in order to get accurate energy models.

3.3.2. Mobile Device Cloudlet Computation. When a mobile device chooses computing its task to the mobile device cloudlet, the input data can be transmitted directly to the nearby mobile devices which have sufficient resources (memory, battery level, processing power). The mobile device user would incur the extra overhead in terms of time and energy for transmitting the computation input data to the mobile device cloudlet. With respect to minimizing execution time, offloading becomes worthy if the local execution time (t_k^{local}) exceeds the offloading execution times ($t_k^{offload}$) which is the sum of remote execution time (t_k^{remote}) plus uploading data time (t_k^{upload}) plus downloading data time (t_k^{down}). Also, $t_k^{offload}$ should be less than the task delay deadline t_k^{max}.

$$t_k^{local} > t_k^{offload}. \tag{3}$$

$$t_k^{offload} = t_k^{remote} + t_k^{upload} + t_k^{down} \tag{4}$$

$$t_k^{remote} = \frac{D_k}{F_{server}} \tag{5}$$

$$t_k^{upload} = \frac{D_{send}}{r_{upload}} \tag{6}$$

$$t_k^{download} = \frac{D_{receive}}{r_{down}} \tag{7}$$

where F_{server} is the compute speed of the mobile device in the cloudlet hosting the task T_k in MIPS. The variables D_{send} and $D_{receive}$ denotes the sent and received data size in bytes respectively. The variables r_{upload} and r_{down} denotes the upload and download data rates in bits per second. The total offloading time to the cloudlet where F_{server} is the computation ability of the devices in the cloudlet. The total energy consumed by the mobile device via offloading to the cloudlet can then be calculated as

$$e_k^{offload} = \frac{D_k}{F_{server}} P_{idle} + \frac{D_{send}}{r_{upload}} P_k^{upload} + \frac{D_{receive}}{r_{upload}} P_k^{down} \tag{8}$$

where, P_{idle}, P_k^{upload}, P_k^{down} denotes, the idle power consumed by mobile device k during remote execution, the power consumed during uploading D_{send} and the power consumped by k during downloading of $D_{receive}$.

3.4. Problem formulation

The aim is to minimize energy consumption in the mobile device by offloading some application tasks to the mobile device cloudlet. The optimization problem can be formulated as

$$\min_{\{a_{k,m}\}} (a_{k,1} \sum_{k=1}^{N_{UE}} t_k^{local} P_a + a_{k,2} (\sum_{k=1}^{N_{UE}} \frac{D_k}{F_{server}} P_{idle} + \frac{D_{send}}{r_{upload}} P_k^{upload} + \frac{D_{receive}}{r_{upload}} P_k^{down})) \tag{9}$$

$$\text{such that, } t_k^{offload} \leq t_k^{max}, \ k \in \mathcal{K} \tag{10}$$

where contraint (10) ensures that the delay constraint of task T_k are met.

3.5. MECH Framework Flowchart

The flow chart in Fig. 4 shows operation of the offloading manager which start with the arrival of a task, T_k. If local execution time is less than the maximum delay tolerable and the battery $SoC \geq 20\%$, the task is executed

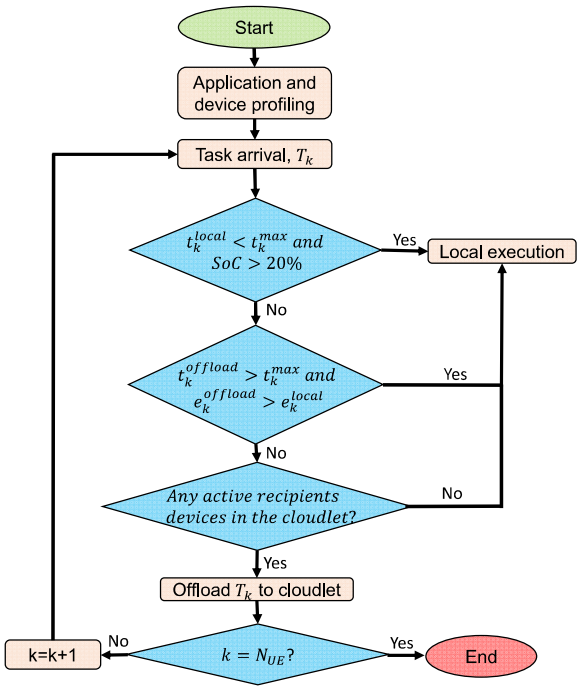

Figure 4. The MECH framework flow chart.

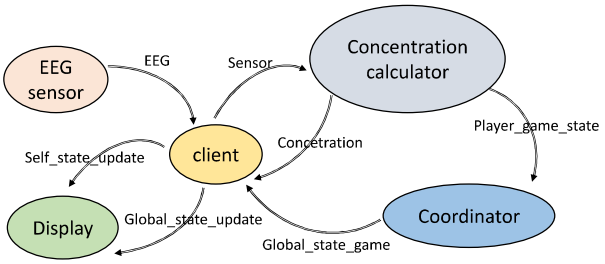

Figure 5. Application model for EEG game [14].

TABLE 1. DESCRIPTION OF INTER-MODULE SETTINGS [14].

Task type	CPU length(MIPS)	Size(bytes)
EEG	2000	500
sensor	3500	500
player_game_state	1000	1000
concentration	14	500
Global_state_game	1000	1000
Global_state_update	1000	500
Self_state_update	1000	500

locally. Else if the offloading delay deadlines are met and energy is saved using offloading, the task is offloaded to the cloudlet in case there are active recipients in the directory service. The next task T_{k+1} then follows the same order in the flow chart.

4. Simulation and Results

4.1. Simulation Settings

The proposed MECH framework is compared with the energy efficient computation offloading (EECO) mechanisms for MEC in [2] and the CANDIS framework in [11]. The application to be considered is the electroencephalogram (EEG) tractor game in [14] that involves augmented brain-computer interaction. The game is played between four mobile devices which act as clients and these devices do not act as servers/recipients and only one device among the four is considered for results analysis. Fig. 5 shows the application model of the EEG tractor game from a graph, $G = \{V, E\}$, where V is a set of vertices denoting the application task modules/classes and E is the edges which denotes data dependencies. EEG module is the sensor that send EEG signals to the client. The client is the mobile device, the display is also in the mobile device. The concentration calculator determines the brain state of the user from the sensed EEG signal values and calculates the concentration level. This module informs the client module about the measured concentration level so that the game state of the player on the display can be updated. Coordinator works at the global level and coordinates the game between multiple players that may be present at geographically distributed locations. The coordinator continuously sends the current state of the game to the client module of all connected users.

The simulation was performed using iFogSim simulation tool [14] which is an extension of cloudsim simulator. The iFogSim is a toolkit for modelling and simulation of resource management techniques in the internet of things and MEC. Table 1 shows the settings used for the mobile application considered.

4.2. Results Evaluation

Fig. 6 shows how the increase in the number of users within the network varies with the energy consumed in the client mobile device. The figure shows that the energy consumption of local computation is constant since local computation is not affected by the number of users in the system. For the EECO scheme, as the number of users increases, the energy consumption also increases because, the EECO scheme offload tasks to a remote MEC server in the BS location, where mobile devices share the bandwidth and the higher the number of users, the less the bandwidth allocated to each user causing poor transmission rate. The poor or limited transmission rate results in high energy consumption in the client mobile device. For the CANDIS and the MECH schemes, as the number of users increases in the system, the energy consumption in the client device decreases because, more devices means the cloudlet is much larger with more active users being able to act as recipient and execute the tasks from the client in a parallel fashion.

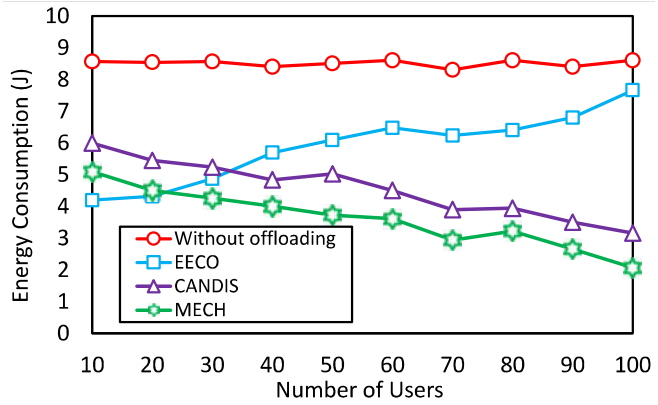

Figure 6. Number of users versus total power consumption on the mobile device.

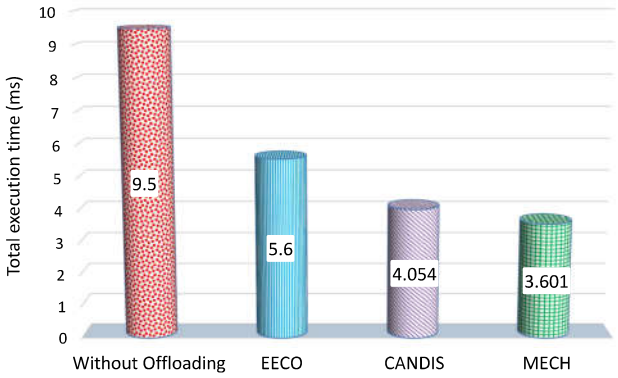

Figure 7. Average response or total execution time.

On average, the proposed scheme out perfoms the scheme without offloading, the EECO and the CANDIS by 58%, 61% and 21% respectively.

Fig. 7 shows average total execution time ($t_k^{offload}$ or t_k^{local}) for all the schemes which was calculated using (3) and (4). On average, the scheme without offloading, the EECO, CANDIS and the proposed MECH takes 9.5ms, 5.6ms, 4.1ms and 3.6ms to execute tasks respectively. The MECH outperforms the scheme without offloading, the EECO and CANDIS by 62%, 37% and 12% respectively. The scheme without offloading performs poorly with high execution time of 9.5ms because there are no remote servers to execute tasks for the mobile device hence local execution is chosen which takes a lot of time to finish execution. The EECO scheme also has high execution time compared to the proposed MECH scheme because in EECO, more delays are inured when offloading the tasks far away at the MEC server in the BS. The CANDIS scheme has slighlty higher execution time of 4.1ms compared to 3.6ms for the proposed MECH scheme because, in MECH, tasks are computed in parallel in the cloudlet.

5. Conclusion

In this paper, a framework called MECH was proposed for saving energy and improving execution time in the mobile devices. In MECH, an application from the mobile device is partitioned into modules/tasks which are offloaded and executed in a mobile device cloudlet while taking into consideration the transmission cost and delay. Simulation results show that the proposed scheme is energy efficient and reduce task execution time. For future considerations, mobility of users will be considered and also the framework will be extended for health and emergency situations.

References

[1] P. T. Joy and K. P. Jacob, "Cooperative caching framework for mobile cloud computing," *arXiv preprint arXiv:1307.7563*, 2013.

[2] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[3] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, and T. P. Jung, "Augmented brain computer interaction based on fog computing and linked data," in *Intelligent Environments (IE), 2014 International Conference on*. IEEE, 2014, pp. 374–377.

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE, 2016, pp. 1–8.

[5] Y. Yu, "Mobile edge computing towards 5g: Vision, recent progress, and open challenges," *China Communications*, vol. 13, no. 2z, pp. 89–99.

[6] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[7] L. Chen, S. Zhou, and J. Xu, "Energy efficient mobile edge computing in dense cellular networks," *arXiv preprint arXiv:1701.07405*, 2017.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[9] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 6.

[10] A. E.-H. G. El-Barbary, L. A. El-Sayed, H. H. Aly, and M. N. El-Derini, "A cloudlet architecture using mobile devices," in *Computer Systems and Applications (AICCSA), 2015 IEEE/ACS 12th International Conference of*. IEEE, 2015, pp. 1–8.

[11] S. Schildt, F. Busching, E. Jorns, and L. Wolf, "Candis: heterogenous mobile cloud framework and energy cost-aware scheduling," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. IEEE, 2013, pp. 1986–1991.

[12] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using mapreduce," DTIC Document, Tech. Rep., 2009.

[13] F. Büsching, S. Schildt, and L. Wolf, "Droidcluster: Towards smartphone cluster computing–the streets are paved with potential computer clusters," in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 2012, pp. 114–117.

[14] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *arXiv preprint arXiv:1606.02007*, 2016.