Mobility and Execution Time Aware Task Offloading in Mobile Cloud Computing

https://doi.org/10.3991/ijim.v16i15.31589

J. Arockia Mary[1(✉)], A. Aloysius[2]
[1]Department of Computer Applications, Holy Cross College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India
[2]Department of Computer Science, St. Joseph's College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India
jarockia79@gmail.com

Abstract—Nowadays, mobile devices perform almost all tasks that can be performed by a computer but empties the battery and consumes memory. It is not necessary to execute the tasks on mobile devices; instead, it is executed in the far-away cloud. To save the battery energy, the tasks are offloaded and hopped through several access points to reach the cloud and executed which increased the execution time of the task. Therefore, to save the execution time and energy, the tasks are offloaded to a nearby cloudlet and as the device moves, the cloudlet and mobile device are disconnected. The mobile device is connected to the next cloudlet, whilst the offloaded tasks are partially executed in the previous cloudlet to VM migrate to the new cloudlet. The previous cloudlet examined the remaining execution time of the task. If it is less than the connection time, the task is finished and the result is transferred to the new cloudlet; otherwise, the task is offloaded to the new cloudlet. It is seen that the mobility and execution time aware task offloading model reduces the execution time and power consumption by 21–40% and 26–34% approximately than the existing mobility-aware offloading approach.

Keywords—mobility, task execution, offloading, VM migration, cloudlet, power, energy

1 Introduction

With the advancement in software and hardware technology, all big data applications are now being executed on smartphones, such as face recognition, 3D games, and weather reports, patient-generated health data [1–2]. However, the processor speed is lower and the memory size is small in smartphones as these resources cannot store additional applications and images. These large size applications in smartphones consume more power and occupya large portion of the memory. Additionally, when executing these applications, the execution time is increased, and the battery of the smartphone is drained more quickly [3], which affects the Quality of Experience (QoE) of the user. Therefore, to reduce the execution time and power consumption of mobile devices, applications or tasks are offloaded from the mobile device to the cloud, which has a larger amount of resources and the execution time is less. Thus, the applications

are offloaded to the cloud, which reduces execution time and saves the battery of the mobile device. In mobile cloud computing, initially, the tasks are offloaded from the mobile device to the cloud [4], which is far away from the user. Popular cloud service providers include Amazon Web Services, Microsoft Azure, Google Cloud, Alibaba Cloud, and Oracle Cloud. For example, Amazon web services launched its first datacenter in Mumbai, India, which has resulted in high latency, lack of network services, and high power consumption. To overcome these drawbacks, Cloudlet [5], which is a small cloud near the access point, was introduced. By offloading the task to the cloudlet, fewer network services are needed, and one hop is needed for the signal to reach the cloudlet from the access point. As the mobile device is moving, it is initially connected to one access point and the task is offloaded for execution to its nearby cloudlet; then, the mobile device is disconnected from the first access point and its cloudlet as the device is moved out of the coverage area of the first access point. The mobile device is connected to the next nearby access point and its cloudlet. If the offloaded task's execution is not completed in the previous cloudlet, the partially executed task must be VM migrated to the nearby cloudlet of an access point where a mobile device is now connected. Different methods have been proposed for mobility-aware task offloading and VMmigration of tasks to the cloudlet [6–10], [12–15], [17–18], [21]. However, mobility-awaretask offloading and migration to the new cloudlet is a critical issue. This article presents the mobility and execution time-aware task offloading and migration method in MCC.

1.1 Contributions of the proposed method

Mobility and execution time-awaretask offloading and migration to virtual machines is an emerging area, where these tasks are offloaded to reduce the execution time and battery life of a mobile device. If there is a disruption in the connection between the mobile device and the cloudlet, QoS parameters such as execution time, response time, and battery life are affected. This study proposes a mobility and execution time aware task where, even though the connection is disrupted between a mobile device and the cloudlet, the task is executed fully in different cloudlets while providing the best service to the mobile users. The offloaded task, which is partially executed in one cloudlet and continues the remaining execution in other cloudlets is a critical issue. To solve this issue, the mobility and execution time aware task offloading and dynamic migration of tasks to the new cloudlet is considered in this study.

The contributions of this study are:

1. A mobility-aware task offloading method is proposed. By predicting the mobility and signal strength of the device, the task was offloaded to the new cloudlet.
2. The execution time-aware dynamic migration method is proposed. Depending upon the predicted remaining execution time of the task, the task is either VM migrated to the new cloudlet or continues the execution in the current cloudlet.
3. In these two proposed methods, the execution time of the task and the power consumption of a mobile device are calculated and compared with the existing mobility-aware offloading model. The proposed model was implemented using VC++ in Microsoft Visual Studio IDE with a rented real cloud.

2 Related works

Several articles have been published on Mobile Cloud Computing (MCC). In MCC, the task is offloaded to the cloud and cloudlets. In [6], authors proposed a method in which the offloading decision was taken dynamically by predicting the user's track using GPS and a greedy searching algorithm to select the best access point and cloudlet. Authors in [7] proposed a new architecture to schedule a task by predicting mobility based onthe history of the mobility pattern of the user. A new architecture to sort cloudlets based on three properties: latency, bandwidth, and distance between the mobile device and the Cloudlet are proposed in [8], where the topmost cloudlet was chosen to offload the task from a mobile device whose speed varies from 2 m/s to 10 m/s. The history of the access points visited by the user was sorted, whereby the most visited access point was selected, and the best cloudlet was selected to offload the task using a genetic algorithm in [9].

In [10], a new architecture is proposed to execute big data tasks by predicting the speed and direction of a mobile device. The computing resources are allocated by the cloud service providers to the clients in order to increase their profit at the same time reduce the prices offered by the clients to acquire the computing resources in a method [11]. The scheduling method sort the users by determining the duration of the associativity of the user to the particular access point's signal range and the best user is allowed to offload the task to the cloudlet, and the task is migrated from the overloaded cloudlet to another cloudlet in [12].

In [13], a scheme is proposed to address seamless offloading using software-defined networking (SDN) and remote cache. This scheme is divided into three layers: the mobile environment, network infrastructure, and cloud infrastructure layers. Mobility is managed by two Open Flow mobility access gateways and a local mobility anchor, reducing response time, energy, and cost. In [14], a method is designed to consider the mobility and context of a task and then offload it uses a fault-tolerance mechanism to find the checkpoint of the task from where it is offloaded and Mobility is defined using the user's history and the Markov chain method to forecast mobility to reduce the execution time of a task, the response time, and the cost.

Architecture [15] consisting of client-side and cloud side components uses the decision engine divides the tasks into lightweight (local) and heavy-weight (offloaded) tasks to offload from the client-side component to any one of the three cloud servers: ad-hoc cloud, cloud server, and cloudlet but this architecture finds failed cloud servers and there are other network failures. In [16], the best four scheduling algorithms are found to reduce the packet losing from mobile nodes but this article does not propose any new method. In the architecture [17], a new mobility-aware adaptive task offloading algorithm is used to reduce the response time. The proposed method is a novel method based on the comparison of existing mobility-aware offloading methods, as shown in Table 1, which considers the migration of the offloaded task from one cloudlet to another cloudlet by determining the signal strength, mobility of the mobile device, and execution time of the offloaded task.

Table 1. Existing and proposed mobility aware task offloading methods and their procedures

Contribution Details	Existing Methods			Proposed Method
	Music [17]	Mobility-aware task delegation model in mobile cloud computing [18]	MAGA [9]	
Contribution procedure	Used two-tier architecture with cloud	Cloudlet and cloud are used. VM migration is done from one cloudlet to another cloudlet	Cloudlet and cloudare used. VM migration is not considered	Cloudlet is used. Determination of VM migration of task from one cloudlet to another cloudlet is considered based on the execution time of the task
QoS factors Addressed	Power and delay	Power and execution time	Power and execution time	Power and execution time

3 Proposed task offloading method

The proposed task offloading work is presented in algorithm 1. Initially, the task is offloaded from a mobile device to a nearby cloudlet. When the mobile device is moving, it loses its connection with the current access point and its nearby cloudlet, and the device is connected to the next access point. In step 5, the cloudlet determines the speed of the device and predicts the time taken by the device to cross a distance of 1000m. The speed of the device S_k was found using equation 1. Table 2 shows the notations used in this method.

$$s_k = \frac{d_k}{t_k} \tag{1}$$

Where *d* denotes the distance traveled by the k^{th} device in meters and *t* denotes the time taken by the k^{th} mobile device in seconds. Once the speed of the device is found, the cloudlet C_k predicts the time taken by the device to cross a distance of 1000m for the calculated speed. The cloudlet already stores different periods for different speeds to reach a distance of 1000m, meaning that signal strength of the access point is usually high, after which, the signal strength is reduced [19]. At this low signal strength, the connection will be cut between the mobile device and the access point, and it is shown that the device starts to connect with the next nearby access point using the new access point's pilot signal [20]. The pilot signal is the strongest of the access point used to connect with the mobile device. Several access points are near the mobile device, but the mobile device connects with the access point whose pilot signal is stronger [21]. When the mobile device is connected to a new access point, the device is automatically associated with its nearby cloudlet.

If the task is still executed, then the cloudlet checks the distance of the device, and the mobile device sends its signal strength. If the signal strength is between −107dBm–−110 dBm and the distance is approximately 1000m i.e. the device is moving near the edge of the coverage area of the current access point and is ready to

connect to the next access point. At this point, the cloudlet determines the remaining execution time of the task. In this case, the execution time of the task is calculated using equation 2, where T_{ex} represents the execution time of the task, *Ic*, $\frac{Cl}{I}$ and *Ct* represent the instruction count of a task, Clocks per instruction, and clock time of the virtual machine, respectively.

If the remaining execution time of the task is less than the propagation time, then the task is executed within the old cloudlet and the result is sent to the new cloudlet and then to the mobile device [22]. In this case, the offloading time $T_{off\text{-}1}$ is calculated using equation 3, where T_{propg} represents the propagation time, which is the connection time between the two devices and is used twice as the first and last terms. The variables T_{upl}, T_{dwl}, and T_{ex} represent the uplink time, downlink time, and execution time of the task, respectively [23]. The power, $T_{pow\text{-}1}$, consumed by the mobile device is calculated using equation 4 in case 1. The variables P_{idl}, P_{se}, and P_{re} represent the power consumption of the mobile device during its idle time, power consumption of the device when sending the data, and power consumption of the device when receiving the data, respectively.

$$T_{ex} = Ic \times \frac{Cl}{I} \times Ct \tag{2}$$

$$T_{off\text{-}1} = T_{propg} + T_{upl} + T_{dwl} + T_{ex} + T_{propg} \tag{3}$$

$$T_{pow\text{-}1} = P_{idl} \times T_{propg} + P_{se} \times T_{upl} + P_{re} \times T_{dwl} + P_{idl} \times T_{ex} + P_{re} \times T_{propg} \tag{4}$$

The distance between the access point and the mobile device is calculated using the great-circle distance between two points on the surface of the sphere. Given that the earth is a sphere, as shown in Figure 1. This distance can be easily calculated using the Haversine formula, given below as equation 5. The distance between the access point and the mobile device is denoted as 'd', and the mobile device location is represented by latitude $\phi 1$ and longitude $\lambda 1$. The location of the access point is also represented by latitude $\phi 2$ and longitude $\lambda 2$.

$$d = 3963.0 \times \text{across}[\sin(\phi 1) \times \sin(\phi 2) + \cos(\phi 1) \times \cos(\phi 2) + \cos((\lambda 1) - (\lambda 2))] \tag{5}$$

Here, the radius of the Earth is 3963 km, the mobile device location is B, and the access point location is A. The distance between A and B is the distance between the sphere. A cloudlet was placed near the access point and the circle represents the coverage area of the access points. When the mobile device is about at 1000m, it attempts to connect with the next access point.

Table 2. List of notations

Notation	Description
S_k	Speed of the device
D	Distance traveled by the k^{th} device in meters
T	Time is taken by the k^{th} mobile device in seconds.
C_k	Cloudlet k
T_{ex}	The execution time of the task,
Ic	Instruction count of task
$\frac{Cl}{I}$	Clocks per instruction
Ct	Clock time of the virtual machine
$T_{off\text{-}1}$	Offloading time
T_{propg}	Propagation time
T_{upl}	Uplink time
T_{dwl}	Downlink time
$T_{pow\text{-}1}$	The power consumed by the mobile device
P_{idl}	Power consumption of the mobile device during its idle time
P_{se}	Power consumption of the device when sending the data
P_{re}	Power consumption of the device when receiving the data
Φ1	Latitude of mobile device location
λ1	Longitude of mobile device location
Φ2	Latitude of the access point
λ2	Longitude of the access point
T_{Tran}	The transmission time of the cloudlet
RTS	Remaining task size
Bw	Bandwidth
Tot_{ex}	The total execution time of the VM migrated task
T_{ofvn-2}	The offloading time of the VM migrated task
T_{pwvn-2}	Power required by the mobile device to offload and execute the task
$T_{pvm\text{-}k}$	Power consumption required by the mobile device to connect with a cloudlet-k
Tot_{pvmexk}	Power consumption required by the mobile device to execute the task in a cloudlet-k
T_{pwvn-2}	Power required by the mobile device to offload and execute the task in all the cloudlets
C_k	K^{th} cloudlet

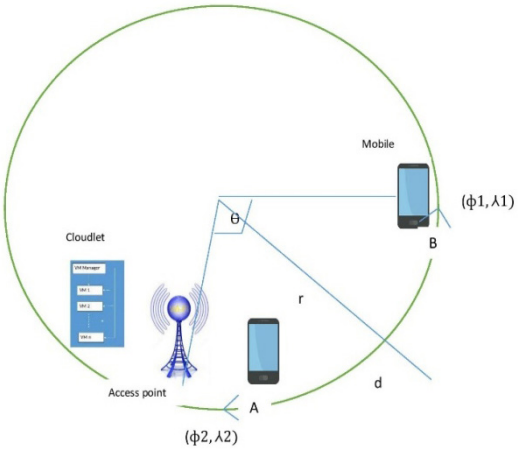

Fig. 1. Distance calculation on the surface of sphere

In case1, the task is completed in the first cloudlet and sends the result to the mobile device from the second cloudlet and no breakup time was required. As soon as the signal strength is reduced, the mobile device is connected to the second cloudlet, and the first cloudlet immediately sends the result to the second cloudlet. The transmission time between the first and second cloudlets is the propagation time only and no task is migrated. Only the result is sent to the mobile device from a nearby cloudlet. Thus, the time and power consumption were reduced.

In Case2, the mobile device is attached with access point A1, its location is within the coverage area of cell1 and its task is offloaded to cloudlet-c1, as shown in Figure 2. As the mobile device moves away from the access point A1, its signal strength decreases, the cloudlet C1 checks the signal strength, and the distancetraveled by the mobile deviceis determined using equation 5. If the signal strength is between −107 dBm to −110 dBm and the distance of the mobile device is greater than 1000m from the access point A1, the mobile device connects with the next access point A2 and its nearby cloudlet C2. Cloudlet C1 checks the remaining execution time of the task using equation 2.

Here, the remaining execution time of the task is greater than the connection time of the mobile device with the next access point A2. The transmission time of the remaining task T_{Tran} from cloudlet C1 to new cloudlet C2 was calculated using equation-6. In equation 6, the remaining task size (RTS) is divided by the bandwidth (Bw) used by cloudlet C1 to transmit the remaining task. The remaining task is VM migration from C1 to C2. If the mobile device is moving away and the task is not completed in its execution in C2, then the mobile device must connect with the next access point-A3. Therefore, the remaining task was offloaded to the new cloudlet C3 and this process continues until the task is complete. Finally, the result is sent from the last cloudlet to the mobile device and the total execution time of the VM migrated task in the different cloudlets is found using equation 7. The offloading time of the VM migrated task in equation 8 is the summation of the offloading time of task in n cloudlets and summation

of transmission time of the mobile device in n cloudlets and summation of the execution time of the task in all the n cloudlets obtained using equation 8.

$$T_{Tran} = \frac{RTS}{Bw(C)} \tag{6}$$

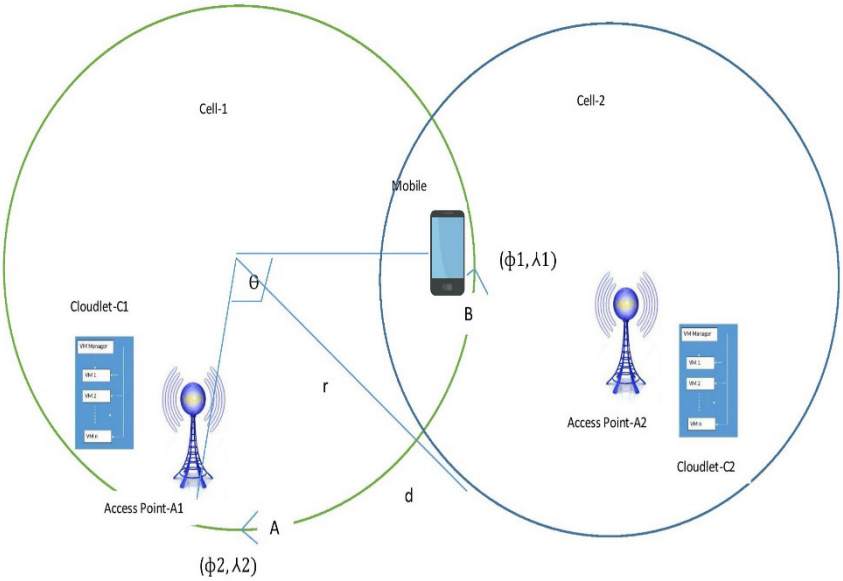

Fig. 2. VM migration of task

$$Tot_{ex} = \sum_{k=1}^{n-1} T_{Trank} + \sum_{k=1}^{n} T_{exk} \tag{7}$$

$$T_{ofvn-2} = \sum_{k=1}^{n-1} T_{ofvn-k} + \sum_{k=1}^{n-1} Tot_{exk} \tag{8}$$

In equation 8, the first summation represents the offloading time taken by the mobile device to each cloudlet upto n cloudlets. The offloading time for a single cloudlet is obtained from equation 3 which is obtained from case1 where the task finished the execution in the first cloudlet, and the result was obtained in the second cloudlet. Each term is obtained from equation 3 without the last term propagation time. Therefore, the modified offloading time is found for the kth cloudlet using equation 9. The second term in equation 8 is the summation of the execution time of the task in all n cloudlets, which is obtained from equation 7

$$T_{ofvmk} = T_{propgk} + T_{uplvmk} + T_{dwlvmk} \tag{9}$$

$$T_{pvm\text{-}k} = P_{idlk} \times T_{propgk} + P_{sek} \times T_{uplk} + P_{rek} \times T_{dwlk} \tag{10}$$

$$\mathrm{Tot}_{\mathrm{pvmexk}} = \mathrm{P}_{\mathrm{idlk}} \times \mathrm{T}_{\mathrm{Trankk}} + \mathrm{P}_{\mathrm{idlk}} \times \mathrm{T}_{\mathrm{exk}} \tag{11}$$

$$T_{pwvn-2} = \sum_{k=1}^{n-1} T_{\mathrm{pvm-k}} + \sum_{k=1}^{n-1} Tot_{\mathrm{pvmexk}} \tag{12}$$

Similarly, the power required by the mobile device to offload and execute the task in all n cloudlets is found using equation 12. This equation is the summation of the power consumption required by the mobile device to connect with n cloudlets, which is the first term in the equation obtained from equation10 and summation of power consumption required by the mobile device to execute the task in n cloudlets, which are obtained from equation 11 and it is the second term in this equation.

Algorithm 1. **Proposed code offloading method using VM migration**

1. Start
2. A mobile device offloads its task to its nearby cloudlet C_k where k = 1
3. The cloudlet C_k starts execution
4. While k < n–1
5. The cloudlet C_k predicts the time taken by the mobile device to reach a distance of 1000m from the current access point.
6. The cloudlet C_k checks whether (distance > 1000m) and (mobile device's signal strength < (–107dbm to –110 dBm)).
7. if (the remaining execution time of the task < connection time) then the task continues its execution and send the result to the next cloudlet C_{k+1} else
8. The cloudlet C_k offload the remaining task to the cloudlet C_{k+1} and the task resumes its execution in C_{k+1}
9. End while
10. C_n sends the result to the mobile device
11. End

4 Performance analysis of proposed algorithm using real cloud servers

Real cloud servers were rented from cloud service providers and used to evaluate the performance of the proposed methods. The I2k2 Network Company is a real cloud service provider that offers private, public, and hybrid cloud servers. This server was-rented with the configuration of the G8 series and 1X Intel Hexa Core Xeon processor ES5-2620, and other details are given in Table 3. The proposed method is written in VC++ and executed on the cloud server. Mobility nodes with different speeds and task sizes were provided in the program and were executed for 10 iterations.

Table 3. Cloud server configurations

IBM/Super/HP DL 160 G8 Series
1 X Intel Hexa Core Xeon Processor E5-2620
15 M Cache, 2.0 GHz, 7.2 GT/s Intel QPI, 2x Gigabit Ethernet Card
6 Cores || 12 Threads
Integrated SAS RAID 0,1,5
8 GB Fully Buffered DDR3 ECC Memory
2 X 1 TB SATA || 7200 RPM HDD
2000 GB Monthly Data Transfer
FREE Set-Up!!, 5 IP Addresses
24x7 Technical Support on Phone/Email/Chat
Free Remote Reboot, Complete Remote Access
Tier 3 Data Center in Delhi NCR

In the proposed method, all types of velocities were stored in the cloudlet starting from a normal walking speed of 1.8 km/h to a typical vehicle speed of 18 km/h. Once the mobile device was connected to the cloudlet, the cloudlet determined the velocity and determined the time taken to reach 1000m distance from the access point. Figure 3 shows the graphs that compare the offloading time and power consumption in the proposed task offloading method with the existing mobility-aware task delegation model MATDM and mobility-aware service allocation method MuSIC in case1. These two methods were chosen because they are mobility-aware task offloading models. The time consumption was calculated in milliseconds (ms), and the power consumption in milliwatts (mW). The experiments with rented cloud show that our task offloading method is compared with MuSIC and reduces time consumption from approximately 77–85% and power consumption from 40–85%. Similarly, with MATDM, the proposed task offloading method reduces time consumption by approximately 17–21% and power consumption by 10–25%.

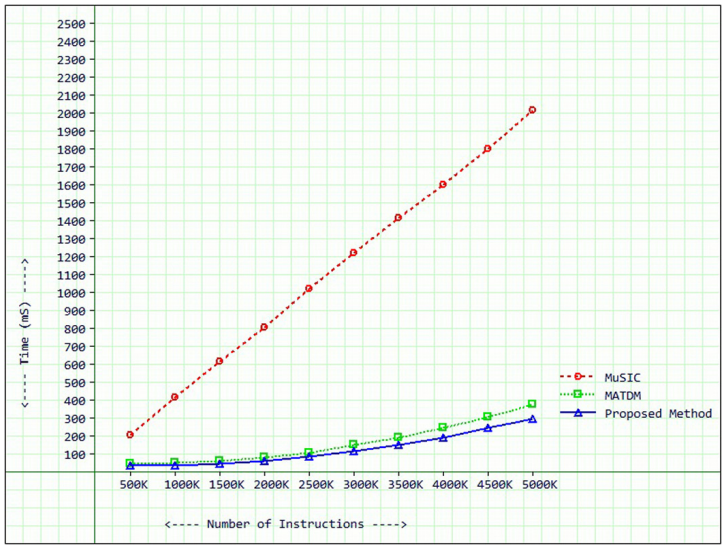

Fig. 3a. Offloading time consumption of the proposed task offloading method in graph

Number of Instructions	MuSIC (mS)	MATDM (mS)	Proposed Method (mS)
500000	208	45	36
1000000	419	50	38
1500000	619	64	47
2000000	808	80	62
2500000	1022	109	88
3000000	1224	150	116
3500000	1419	192	150
4000000	1601	246	194
4500000	1804	308	246
5000000	2018	376	298

Fig. 3b. Offloading time consumption of the proposed task offloading method in mS

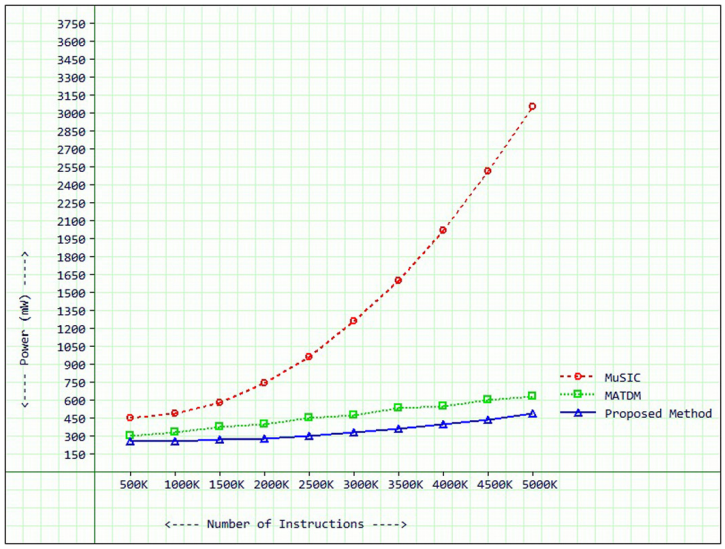

Fig. 3c. Offloading power consumption of the proposed task offloading method in graph

Number of Instructions	MuSIC (mW)	MATDM (mW)	Proposed Method (mW)
500000	456	305	262
1000000	489	332	259
1500000	582	379	270
2000000	743	400	284
2500000	962	450	305
3000000	1262	476	336
3500000	1600	533	360
4000000	2022	553	404
4500000	2513	601	440
5000000	3058	630	491

Fig. 3d. Offloading time consumption of the proposed task offloading method in mW

Figure 4 shows the graphs that compare the offloading time and power consumption in the proposed task offloading VM migration method, as shown in Figure 4 with the existing mobility-aware task delegation model MATDM and mobility-aware service allocation method MuSIC. The experiments with real cloud rented for three months show that our task offloading method is compared with MuSIC and reduces time consumption from approximately 97–98% and power consumption from 22–71%. Similarly, with MATDM, the task offloading method reduces time consumption by approximately 2–50% and power consumption by 8–26%.

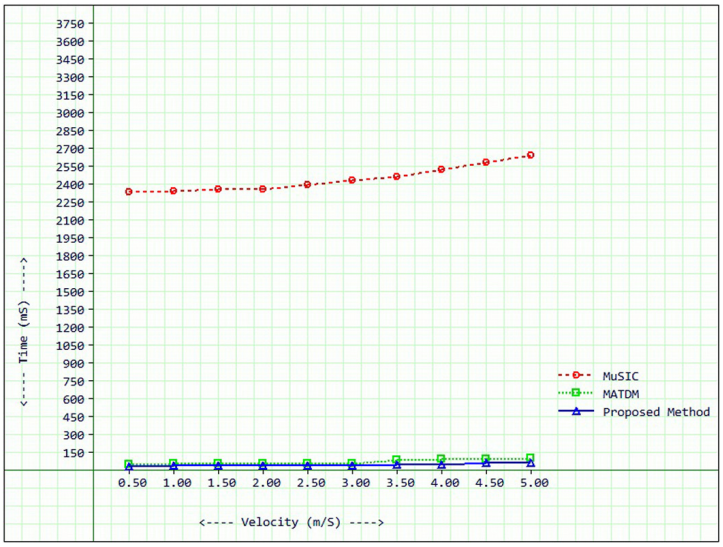

Fig. 4a. Offloading time consumption of the proposed task offloading method in graph

Velocity (m/S)	MuSIC (mS)	MATDM (mS)	Proposed Method (mS)
0.5	2333	50	34
1	2345	53	38
1.5	2356	54	39
2	2360	56	38
2.5	2396	53	44
3	2434	54	40
3.5	2464	84	51
4	2527	92	50
4.5	2580	90	61
5	2643	102	65

Fig. 4b. Offloading time consumption of the proposed task offloading method in mS

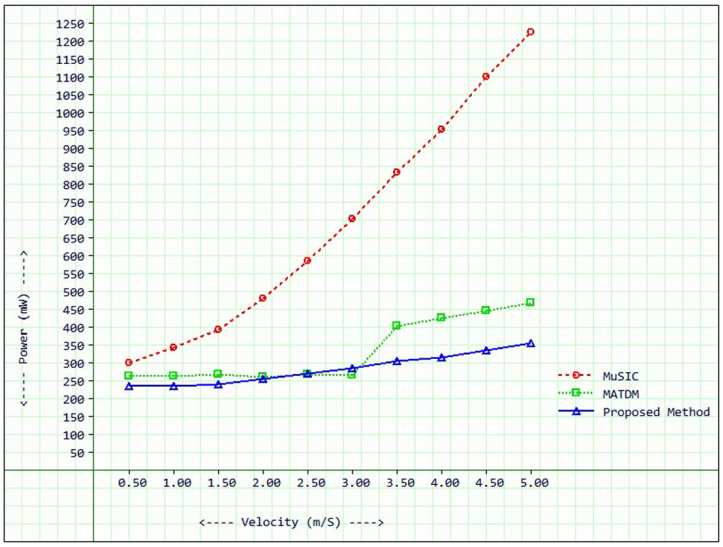

Fig. 4c. Offloading power consumption of the proposed task offloading method in graph

Velocity (m/S)	MuSIC (mW)	MATDM (mW)	Proposed Method (mW)
0.5	300	263	236
1	343	264	236
1.5	394	268	241
2	480	260	255
2.5	585	268	271
3	704	266	286
3.5	834	404	305
4	954	426	317
4.5	1101	445	336
5	1227	469	356

Fig. 4d. Offloading time consumption of the proposed task offloading method in mW

5 Conclusions

Mobility and execution time-aware offloading in mobile cloud computing is proposed in this article. It considers mobility as well as execution time to offload the task to the respective cloudlet. This proposed method uses VM migration and is compared with existing mobility-aware music and MATDM methods and reduces the execution time of the task and power consumption of the mobile device. This study is suitable for executing larger tasks and reducing the execution time. Similarly, when the velocity is increased, the power consumption of the mobile device is reduced. When the mobile device is at the edge of the coverage area of the access point, the decision is taken perfectly to connect with the next access point and the next cloudlet based on the signal strength and performs VM migration based on the remaining execution time of the task. There is no dilemma as to whether to stick with an existing access point and its cloudlet or to the new one. This is because the remaining execution time of the task plays a major role in determining VM migration. This proposed method reduces network resource consumption because small-sized mobile tasks are executed in a single cloudlet by avoiding the connection of unnecessary cloudlets. This proposed method has been experimented with in real cloudlets rented from i2k2 network solutions. Therefore, the results were absolute. In the future, the same method can be tested with 5G technology, where big data and signal strength are higher.

6 References

[1] Juhi M. Purswani et al., "Big Data From Small Devices: The Future of Smartphones in Oncology", *Seminars in Radiation Technology*, Vol. 29, no. 4, pp. 338–347, 2019. https://doi.org/10.1016/j.semradonc.2019.05.008

[2] Orasa Sirasakamol, Pakinee Ariya, Wanvimol Nadee, Kitti Puritat," Development of a Mobile-Healthcare Application for Safety and Prevention in Emergency Assistance at Marathon Events: A Case Study in CMU Marathon", *International Journal of Online and Biomedical Engineering,* Vol. 18, no. 6, pp. 65–81, 2022. https://doi.org/10.3991/ijoe.v18i06.29515

[3] Jameel Ali and Majid Altamimi, "Energy Consumption Model for Data Transfer in Smartphone", *Computer Communications,* Vol. 182, Jan., pp. 13–21, 2022. https://doi.org/10.1016/j.comcom.2021.10.014

[4] S. Singh, I. Chana, "Q-aware: Quality of Service-Based Cloud Resource Provisioning", *Computers & Electrical Engineering*, Vol. 47, Oct., pp. 138–160, 2015. https://doi.org/10.1016/j.compeleceng.2015.02.003

[5] Chaogang Tang, Mingyang Hao, Xianglin Wei and Wei Chen, "Energy-Aware Task Scheduling in Mobile Cloud Computing", *Distributed Parallel Databases,* Vol. 36, pp. 529–553, 2018. https://doi.org/10.1007/s10619-018-7231-7

[6] Jiwei Li, Kai Bu, Xuan Liu, and Bin Xiao, "ENDA: Embracing Network Inconsistency for Dynamic Application Offloading in Mobile Cloud Computing", In Proc. Second ACM SIGCOMM Workshop on Mobile Cloud Computing '08, 2013, pp. 39–44. https://doi.org/10.1145/2491266.2491274

[7] Bidoura Ahmad Hridita, Mohammad Irfan, and Md. Shariful Islam, "Mobility Aware Task Allocation for Mobile Cloud Computing", *International Journal of Computer Applications*, Vol. 137, no. 9, Mar., pp. 35–41, 2016. https://doi.org/10.5120/ijca2016908941

[8] M. Akter, F. Zohra, and Amit Kumar Das, "Q-MAC: QoS and Mobility Aware Optimal Resource Allocation for Dynamic Application Offloading in Mobile Cloud Computing", In Proc. ECCE International Conference on Electrical, Computer and Communication Engineering '02, 2017, pp. 803–808. https://doi.org/10.1109/ECACE.2017.7913013

[9] Yan Shi, Shanzhi Chen, and Xiang Xu, "MAGA: A Mobility-Aware Computation Offloading Decision for Mobile Cloud Computing", *IEEE Internet of Things Journal*, Vol. 5, no. 1, Feb., pp. 164–174, 2018. https://doi.org/10.1109/JIOT.2017.2776252

[10] Asma Enayet, Md. Abdur Razzaque, Mohammad Mehedi Hassan, Atif Alamri, and Giancarlo Fortinq, "A Mobility-Aware Optimal Resource Allocation Architecture for Big Data Task Execution on Mobile Cloud in Smart Cities", *IEEE Communications Magazine*, Vol. 56, no. 2, Feb., pp. 110–117, 2018. https://doi.org/10.1109/MCOM.2018.1700293

[11] Ayoub Alsarhan, Bashar Alkhawaldeh, Atalla Fahed Al-Serhan, and Muhsen Alkhalidy, "A Novel Genetic and Intelligent Scheme for Service Trading in IoT Fog Networks", *International Journal of Interactive Mobile Technologies*, Vol. 16, no. 10, pp. 191–209, 2022. https://doi.org/10.3991/ijim.v16i10.30479

[12] Qinglan Peng, Yunni Xia, Zeng Feng, Jia Lee, Chunrong Wu, Xin Luo, Wanbo Zheng, Shanchen Pang, Hui Liu, Yidan Qin and Peng Chen, "Mobility-Aware and Migration-Enabled Online Edge User Allocation in Mobile Edge Computing", In Proc. IEEE International Conference on Web Services '07, 2019, pp. 91–98. https://doi.org/10.1109/ICWS.2019.00026

[13] Warley Junior, Adriano Franca, Kelvin Dias, Jose N. de Souza, "Supporting Mobility-Aware Computational Offloading in Mobile Cloud Environment", *Journal of Network and Computer Applications*, Vol. 94, Sep., pp. 93–108, 2017. https://doi.org/10.1016/j.jnca.2017.07.008

[14] Razie Roostaei and Zeinab Movahedi, "Mobility and Context-Aware Offloading in Mobile Cloud Computing", In Proc. IEEE Intl Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress '07, 2016, pp. 1144–1148. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0176

[15] Abdullah Lakhan, Xiaoping Li, "Mobility and Fault Aware Adaptive Task Offloading in Heterogeneous Mobile Cloud Environments", *EAI Endorsed Transactions on Mobile Communications and Applications,* Vol. 5, no. 16, Jan, 2019. https://doi.org/10.4108/eai.3-9-2019.159947

[16] Hussain Falih Mahdi, Mohammed Hasan Alwan, Baidaa Al-bander, and Aws Zuhair Sameen, "A Comparision of Node Detection Algorithms Over Wireless Sensor Network", *International Journal of Interactive Mobile Technologies*, Vol. 16, no. 7, pp. 38–53, 2022. https://doi.org/10.3991/ijim.v16i07.24609

[17] M. Reza Rahimi, Nalini Venkatasubramanian, and Athanasios V. Vasilakos, "MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing", In Proc. IEEE Sixth International Conference on Cloud Computing '07, 2013. https://doi.org/10.1109/CLOUD.2013.100

[18] Anwesha Mukherjee, DeepsubhraGuha Roy, Debashis De, "Mobility-Aware Task Delegation Model in Mobile Cloud Computing", *The Journal of Supercomputing,* Vol. 75, no. 1, Jan., pp. 314–339, 2019. https://doi.org/10.1007/s11227-018-02729-x

[19] S. Saravanan, P. Sudhakar, "Analysis of Mobile Internet Speed, Signal Strength and FMDH Antenna Design for Improved Internet Speed", *The Journal of Supercomputing,* Vol. 76, no. 6, Jun., pp. 4449–4475, 2020. https://doi.org/10.1007/s11227-018-2382-x

[20] Bo-Chieh Liu and Ken-Huang Lin, "Wireless Location Uses Geometrical Transformation Method With Single Propagation Delay: Model and Detection Performance", *IEEE Transactions on Vehicular Technology,* Vol. 57, no. 5, Sep., pp. 2920–2932, 2008. https://doi.org/10.1109/TVT.2007.912608

[21] Ali Al-Shuwaili, Osvaldo Simeone, Alireza Bagheri, and Gesualdo Scutari, "Joint Uplink / Downlink Optimization for Backhaul-Limited Mobile Cloud Computing with User Scheduling", *IEEE Transactions on Signal and Information Processing over Networks,* Vol. 3, no. 4, Dec., pp. 787–802, 2017. https://doi.org/10.1109/TSIPN.2017.2668142

[22] Qiuheng Zhou and Wei Jiang, "Implementation of open air interface-based real-world channel measurement for evaluating wireless transmission algorithms", in *arxiv:2101.04608*, Cornell University, 2021. https://arxiv.org/pdf/2101.04608.pdf

[23] Yongnam Han, Hang GuBahk and Seungtalk Yang, "CDMA Mobile System Overview: Introduction, Background and System Concepts", *ETRI Journal,* Vol. 19, no. 3, Oct., pp. 83–97, 1997. https://doi.org/10.4218/etrij.97.0197.0301

7 Authors

J. Arockia Mary doing her Doctoral Degree in Computer Science in St. Joseph's College (Affiliated to Bharathidasan University), Tiruchirappalli, Tamilnadu, India. She has more than twelve years of teaching and research experience. She is currently working as Assistant Professor, Department of Computer Applications, Holy Cross College, Tiruchirappalli, Tamilnadu, India. She has published a paper in international journals. She has presented a paper at an international conference. Presently she is doing her research on Mobile Cloud computing. (email: jarockia79@gmail.com)

Dr. A. Aloysius is working as an Assistant Professor in the Department of Computer Science, St. Joseph's College, Trichy, Tamil Nadu, India. He has 22 years of experience in teaching and research. He has published many research articles in national and international conferences and journals. He also presented research articles in the International Conferences on Computational Intelligence and Cognitive Informatics in Indonesia. He has acted as a chairperson for many national and international conferences. His current research area includes cognitive aspects in software design, big data, and cloud computing. (email: aloysius1972@gmail.com)

Article submitted 2022-04-10. Resubmitted 2022-06-07. Final acceptance 2022-06-09. Final version published as submitted by the authors.