J Supercomput (2017) 73:1672–1690
DOI 10.1007/s11227-016-1872-y

Application-aware cloudlet selection for computation offloading in multi-cloudlet environment

Deepsubhra Guha Roy[1] · **Debashis De**[1] · **Anwesha Mukherjee**[1] · **Rajkumar Buyya**[2,3]

Published online: 14 September 2016

Abstract Latency- and power-aware offloading is a promising issue in the field of mobile cloud computing today. To provide latency-aware offloading, the concept of cloudlet has evolved. However, offloading an application to the most appropriate cloudlet is still a major challenge. This paper has proposed an application-aware cloudlet selection strategy for multi-cloudlet scenario. Different cloudlets are able to process different types of applications. When a request comes from a mobile device for offloading a task, the application type is verified first. According to the application type, the most suitable cloudlet is selected among multiple cloudlets present near the mobile device. By offloading computation using the proposed strategy, the energy consumption of mobile terminals can be reduced as well as latency in application execution can be decreased. Moreover, the proposed strategy can balance the load of the system by distributing the processes to be offloaded in various cloudlets. Consequently, the probability of putting all loads on a single cloudlet can be dealt for load balancing. The proposed algorithm is implemented in the mobile cloud computing laboratory of our university. In the experimental analyses, the sorting and searching processes, numerical operations, game and web service are considered as the tasks to be offloaded to the cloudlets based on the application type. The delays involved in offloading various applications to the cloudlets located at the university laboratory,

✉ Debashis De
dr.debashis.de@gmail.com

[1] Department of Computer Science and Engineering, West Bengal University of Technology, B.F.-142, Sector-I, Salt Lake, Kolkata 700064, West Bengal, India

[2] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

[3] Manjrasoft Pty Ltd, Melbourne, Australia

using proposed algorithm are presented. The mathematical models of total power consumption and delay for the proposed strategy are also developed in this paper.

Keywords Cloudlet · Offloading · AppSpecCloudlet · Power reduction · Delay reduction

1 Introduction

Mobile device suffers from limited storage, limited battery life and limited computing power. To deal with these problems, mobile cloud computing (MCC) is introduced by combining cloud computing and mobile computing [1–5]. Cloud offers virtualization of vast quantity of resources with a distributed computing model providing software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) [1–3]. In MCC the data storage and computation happen inside the cloud and the result is returned to the mobile device. This process is known as offloading [6,7]. Mobile computing is integrated with cloud computing to introduce a new technology mobile cloud computing [7–11]. Cloudlet is an evolution in the field of cloud computing. Cloudlet is defined as multiple computers or a resource-rich computer alone consists of the cache copies of the data stored inside the cloud [12,13]. Use of cloudlet to offload data and computation has alleviated the disadvantage of wide area network (WAN) delay while using cloud. A cloudlet is well connected to the internet. Mobile devices, e.g., mobile phone, tablet, laptop use the cloud services at high bandwidth and low latency through the use of cloudlet.

1.1 Motivation and contributions of proposed work

There may be a case where multiple cloudlets exist near the mobile device, but most users access a particular cloudlet. Therefore that cloudlet becomes overloaded, whereas other cloudlets remain idle. The purpose of offloading is to reduce latency and power consumption. However, if one cloudlet becomes overloaded and others remain idle, the latency is increased. Therefore the quality of service (QoS) as well as quality of experience (QoE) becomes inferior. In such a case proper load distribution does not occur. Our aim is to offer such a strategy which will balance load of the system, reduce the latency as well as energy consumption. If instead of putting all the processing inside a single cloudlet, the applications are distributed among multiple cloudlets, the problem can be resolved. When multiple cloudlets with variations in application processing capabilities are available near a mobile device, then application-specific cloudlet selection is required. In such a scenario, different cloudlets can execute different types of process codes, e.g., numerical operations, sorting and searching operations, game, etc. Our motivation is to propose an application-specific cloudlet selection approach for such a multi-cloudlet environment. The contributions of this paper are:

1. We have proposed a method for selecting cloudlet in multi-cloudlet environment from the perspective of the type of application to be offloaded.

2. The mathematical models of power consumption and delay of the proposed method are proposed.
3. The performance of the proposed strategy is evaluated using experimental results obtained from the cloudlets present in our university laboratory.
4. Using the proposed strategy the total load of the system is balanced, power consumption by the mobile terminals is reduced and latency in offloading application is reduced than the remote cloud-based offloading.

1.2 Organization of rest of the paper

Related works are given in Sect. 2, the proposed cloudlet selection algorithm is given in Sect. 3, the power and delay consumption model of the proposed scheme are developed in Sect. 4. Section 5 presents the experimental results demonstrating the performance of the proposed algorithm and a comparison between the proposed strategy and existing schemes on offloading using cloudlet, and the paper is concluded in Sect. 6.

2 Related work

Offloading applications at low power and low latency is an emerging research area in MCC [6–11]. The use of cloud for reducing the power consumption of mobile devices is discussed in [8]. Different challenges in this area are also discussed in [8]. The process of offloading an application to the cloud is illustrated in [9]. The challenges of application offloading are explored in [9]. The application migration using cloud is described in [10]. The process of resource allocation in an energy-efficient way for the cloud data centers is discussed in [11]. But in these schemes offloading application to cloud causes delay. As the mobile device resides far away from the cloud, the WAN delay occurs. To overcome this difficulty, cloudlet has come [12,13]. Cloudlet acts as an agent between the mobile device and cloud to provide cloud services to the connected mobile device [12]. The advantage of our present work over these existing schemes [4–11] is that in our approach cloudlet has been used for offloading instead of the cloud to reduce the delay.

A large number of cloudlets are allocated and the nearest cloudlet is selected for providing service to the mobile device in [13]. As cloudlet is able to serve limited number of devices, the security is also better in this case with respect to the cloud. For large-scale body area network, a method for gathering data is proposed in [14]. Here cloudlet is used to decrease the power and delay in data collection. A cloudlet platform for augmented reality is proposed in [15]. Here cloudlets can be generated dynamically. A hierarchical storage system based on cloudlet is proposed in [16]. Although these methods have shown the advantage of cloudlet over cloud in various aspects like healthcare, augmented reality, they do not focus on how a cloudlet will be selected in case of multi-cloudlet environment in order to give minimum power and minimum delay. In our present work, we have focused on this area where a cloudlet selection strategy is proposed for multi-cloudlet environment from the perspective of the type of application to be offloaded.

Due to user mobility the optimal resource allocation varies. To solve this problem, two heuristic algorithms are proposed in [17]. Efficient resource management scheme for mobile cloud environment is proposed in [18]. Another resource scheduling approach for cloud environment is proposed in [19]. The scheme contains three phases namely resource matching, resource selection and feedback integration. Mobile storage augmentation is discussed in [20]. A QoS-aware resource scheduling scheme is proposed for managing work load in cloud environment in [21]. Resource management for cloud environment is discussed in [22–24]. The method of partitioning applications at runtime on smart mobile devices, resource utilization is considered in [25]. The process of offloading computations in the cloud environment of mobile ad hoc network is discussed in [26]. An active service migration model is proposed for offloading computations to the datacenter of cloud in [27]. Load balancing in cloud is discussed in [28]. Although several strategies exist for computation offloading to the remote cloud, cloudlet is always a better option as it offers better QoS with respect to low latency and low power. In our previous work, we have proposed a cloudlet selection strategy to achieve low latency and low power in multi-cloudlet scenario [29]. In that case a proxy server has been used for communicating with the cloudlet where offloading occurs. Due to the presence of this proxy, again a delay occurs. In this paper, we have considered the application type while offloading occurs. The available cloudlets are divided into different categories based on their executable application type. By allocating cloudlets based on requested types of applications, the work load is distributed. This in turn balances the load of the system as well as reduces delay.

3 Proposed cloudlet selection strategy

3.1 Three-layer architecture of cloudlet-based offloading

The three-layer architecture of cloudlet-based offloading is pictorially presented in Fig. 1.

Layer 1 contains the mobile devices, layer 2 contains the agent cloudlet and layer 3 consists of the cloud providing IaaS, PaaS and SaaS. Mobile devices of layer 1 request for services to the cloudlet at layer 2. The cloudlet provides service to the requested mobile device if possible. Otherwise it requests for service to the cloud. The cloud at layer 3 receives the request and then sends the result to the cloudlet. After getting result from the cloud at layer 3, the cloudlet at layer 2 sends back the result to the requested mobile device at layer 1.

3.2 Proposed application-specific offloading method for multi-cloudlet environment

In the proposed approach, depending on the application type, the suitable cloudlet is selected for offloading individual task in a multi-cloudlet environment. The multi-cloudlet environment with three cloudlets is shown in Fig. 2.

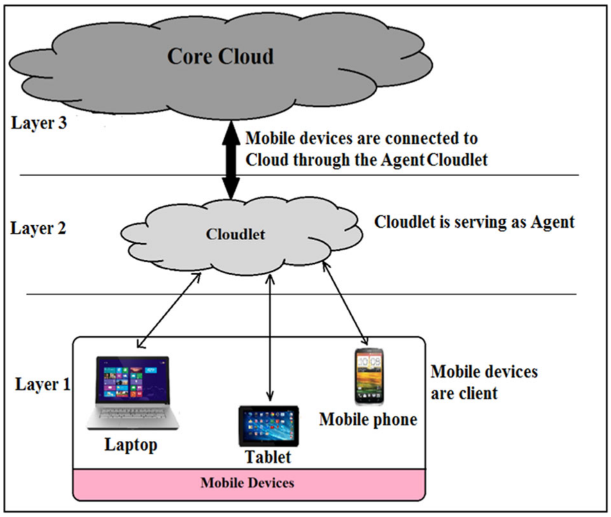

Core Cloud
Layer 3
Mobile devices are connected to Cloud through the Agent Cloudlet
Layer 2
Cloudlet
Cloudlet is serving as Agent
Layer 1
Laptop
Tablet
Mobile phone
Mobile devices are client
Mobile Devices

Fig. 1 Three-layer architecture of cloudlet-based offloading

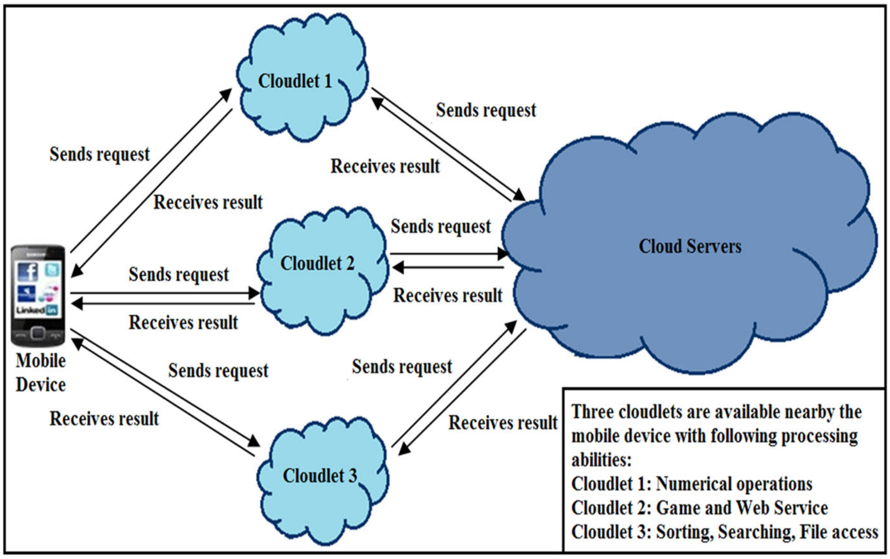


Fig. 2 Multi-cloudlet environment near the mobile device

When a mobile device asks for offloading a task, the power consumption in local execution of the task inside the mobile device is calculated as [29]:

$$P_{\text{local}} = P_{\text{m}}(I/S_{\text{m}}), \tag{1}$$

where P_{m} is the power consumed by the mobile device per instruction execution, S_{m} is the speed of the mobile device and I is the number of instructions to be executed.

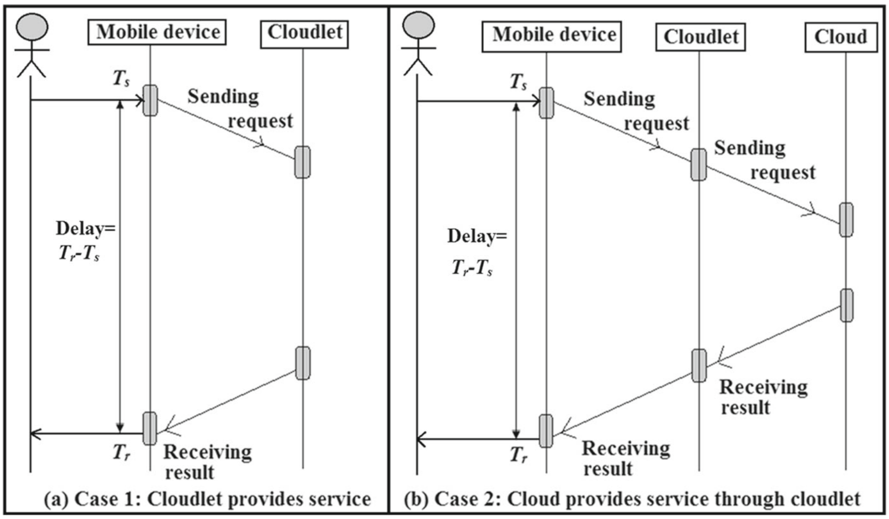


Fig. 3 Offloading delay using cloudlet

The power consumption in offloading the task is calculated as [30]:

$$P_{\text{offload}} = P_{\text{mi}}[(I/S_{\text{clt}}) + (D_{\text{clt}}/S_{\text{pro}})] + (P_{\text{ts}} Da_u/D_u) + (P_{\text{tr}} Da_d/D_d), \quad (2)$$

where P_{mi} is the power consumed by the mobile device in idle state, S_{clt} is the speed of the cloudlet, P_{ts} and P_{tr} are the power required per unit time in sending and receiving data to and from the cloudlet, respectively, Da_u and Da_d are the amount of data transmitted in uplink and downlink, respectively, and D_u and D_d are the uplink and downlink data transmission rate, respectively, D_{clt} is the distance of the cloudlet from the mobile device, and S_{pro} is the propagation speed.

If $P_{\text{local}} < P_{\text{offload}}$, the application is processed within the mobile device. Else the task deadline is compared with the offloading delay. The delay for offloading an application using cloudlet is presented in Fig. 3.

If cloudlet provides service, then the offloading delay is given by:

$$T_{\text{offloadclt}} = (I/S_{\text{clt}}) + [(Da_u/D_u) + (Da_d/D_d)] + (D_{\text{clt}}/S_{\text{pro}}), \quad (3)$$

where (I/S_{clt}) presents the processing delay, (Da_u/D_u) presents the uplink communication delay, (Da_d/D_d) presents the downlink communication delay, and $(D_{\text{clt}}/S_{\text{pro}})$ presents the propagation delay. If cloud provides service through cloudlet, then the offloading delay is given by:

$$T_{\text{offloadcl}} = (I/S_{\text{cl}}) + [(Da_u/D_u) + (Da_d/D_d)] + [(D_{\text{clt}}/S_{\text{pro}}) + (D_{\text{cll}}/S_{\text{pro}})], \quad (4)$$

where D_{cll} is the distance of the cloudlet from the cloud and S_{cl} is the speed of the cloud. Here (I/S_{cl}) presents the processing delay, (Da_u/D_u) presents the uplink communication delay, (Da_d/D_d) presents the downlink communication delay, and $[(D_{\mathrm{clt}}/S_{\mathrm{pro}}) + (D_{\mathrm{cll}}/S_{\mathrm{pro}})]$ presents the propagation delay.

If the task deadline is greater than the offloading delay, then it is checked whether any cloudlet is available nearby. If more than one cloudlet is present, the application type of the task is checked. Based on that application type, the suitable cloudlet is selected. If none of the cloudlets is able to execute that type of application, the task is offloaded to the cloud. If only one cloudlet is present nearby and it is able to execute the task, it is offloaded to that cloudlet. If the deadline of the task is less than the offloading delay and the power consumption of local execution is more than that of the offloading, the user is asked to extend the deadline. If the user refuses to extend the deadline, the battery life of the mobile device is checked. If the current battery life permits the execution of the task, it is locally executed. Otherwise the request is aborted by the system. The proposed application-specific cloudlet selection algorithm is presented in Table 1.

Figure 4 shows the flow diagram of the proposed algorithm.

4 Delay and power consumption in proposed method

4.1 Total delay consumption

In the proposed approach at first the selection of appropriate cloudlet among multiple cloudlets takes place from the perspective of requested application type. After that the application is offloaded to the selected cloudlet. Therefore two delays are involved in the proposed scheme:

(i) Cloudlet selection delay (T_{sc}),
(ii) Offloading delay $(T_{\mathrm{offloadclt}})$.

At first the mobile device broadcasts a message to its nearby cloudlets requesting for their executable application types. After receiving response from the cloudlets, the mobile device stores the information in a table named as Cloudlet Information Table (CIT). The format of CIT is given in Table 2.

As observed from Table 2, a cloudlet may be able to execute more than one type of application. Accessing this table the mobile device selects the suitable cloudlet and offloads the application to that cloudlet.

The cloudlet selection delay considers:

(i) The delay in sending message to the nearby cloudlets (T_{send}),
(ii) The delay in receiving response from the nearby cloudlets (T_{recv}),
(iii) The delay in accessing CIT (T_{access}).

The CIT access delay is given as:

$$T_{\mathrm{access}} = \sum_{i=1}^{N_r} \sum_{j=1}^{n_{\mathrm{sr}}} T_{ij} \tag{5}$$

where T_{ij} is the delay in accessing jth sub-row of i th row, the sub-rows contain the application types executable by ith cloudlet, n_{sr} is the sub-row number containing the

Table 1 Algorithm 1: Proposed application-specific cloudlet selection algorithm

```
Considerations:
  • Available number of cloudlets near the mobile device is $N$.
  • Set of applications executable by these cloudlets are denoted by $A=\{A_1, A_2,....,A_N\}$ where
    $A_i=\{T_{i1}, T_{i2},...,T_{ik}\}$, $1 \le i \le N$ and $k$ is the number of tasks related to application $A_i$.
  • User requests for task $T_j$ belongs to application $A_p$.
  • Power consumption in offloading $T_j$ is $P_{offloadj}$ calculated using Eq. (2)
  • Power consumption in local execution of $T_j$ is $P_{localj}$ calculated using Eq. (1)
  • Deadline of $T_j$ is $D_j$
  • Offloading delay of $T_j$ is $T_{offloadcltj}$ calculated using Eq. (3)
Procedure:
  1:  Start
  2:  A mobile device checks the deadline of the task to be offloaded
  3:  If $P_{offloadj}>P_{localj}$,
  4:      Execute $T_j$ locally inside the mobile device
  5:  Else if $D_j > T_{offloadcltj}$,
  6:      Go to step 7
                          /*Cloudlet Selection*/
  7:      If multiple cloudlets processing different types of application are available near the
          mobile device,
  8:          The application type of the task is checked
  9:          If the application type of the task belongs to the set of applications executable
              by the  available cloudlets i.e. $A_p \in A$,
 10:              The cloudlet with the ability of executing that type of task is selected
                  for offloading
 11:          Else if none of the available cloudlets can execute that type of task,
                  $T_j$ is offloaded to the cloud
 12:          End if
 13:      End if
 14:  Else if $D_j < T_{offloadcltj}$,
 15:      Sends a request to the user asking to extend the deadline
 16:      If the user extends the deadline,
 17:          Go to step 7
 18:      Else if the user refuses to extend the deadline,
 19:          The system checks the current battery life of the mobile device
 20:          If the current battery life of the mobile device permits execution of the
              application, the task is locally executed
 21:          Else the system aborts the request
 22:          End if
 23:      End if
 24:  End if
 25:  End
```

requested application type executable by ith cloudlet and N_r is the row number upto which the table is accessed. When the cloudlet having ability to execute the requested application is found, the process of accessing the table is stopped. As there are N cloudlets, the number of row in the table is N. If the selected cloudlet is at the N_r row, then $1 \le N_r \le N$. If the number of application type executable by the cloudlet at N_r row is n_r, the number of sub-rows corresponding to N_r is n_r. If the requested application is at n_{sr} sub-row, then $1 \le n_{\mathrm{sr}} \le n_r$.

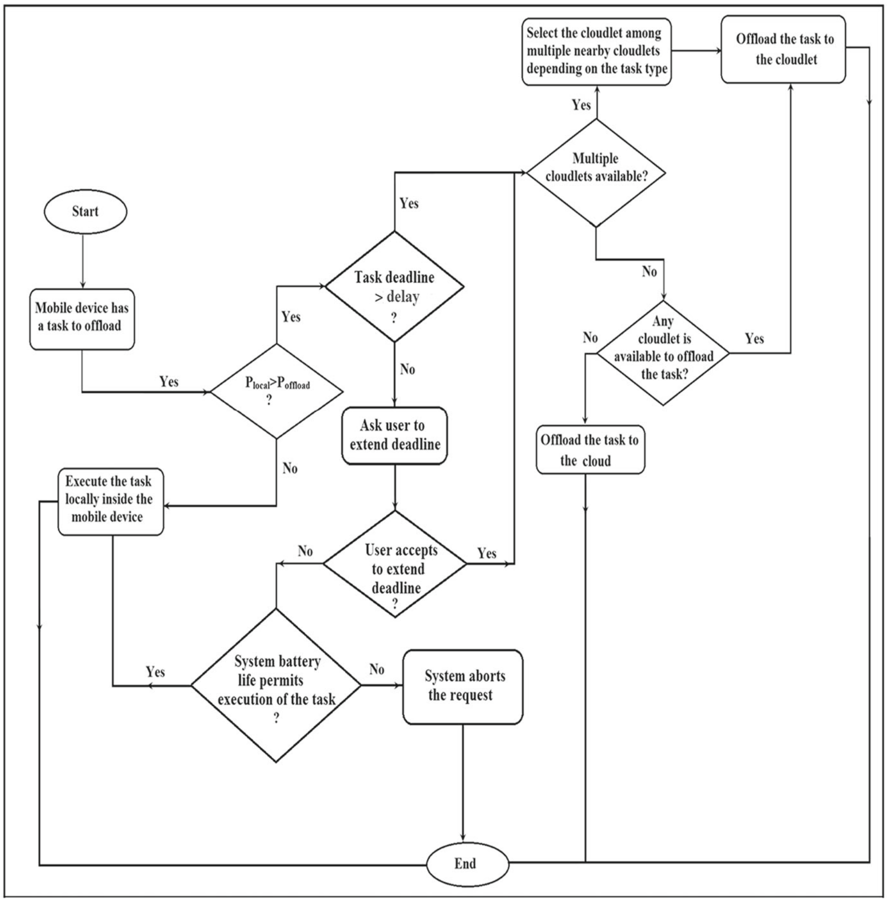

Start
Mobile device has a task to offload
Yes
P_local>P_offload ?
No
Execute the task locally inside the mobile device
Yes
Task deadline > delay ?
No
Ask user to extend deadline
User accepts to extend deadline ?
No
Yes
System battery life permits execution of the task ?
Yes
No
System aborts the request
Multiple cloudlets available?
Yes
Select the cloudlet among multiple nearby cloudlets depending on the task type
Offload the task to the cloudlet
No
Any cloudlet is available to offload the task?
No
Yes
Offload the task to the cloud
End

Fig. 4 Flow diagram of proposed application-specific cloudlet selection strategy

The cloudlet selection delay is given as:

$$T_{\text{sc}} = f(T_{\text{send}}, T_{\text{recv}}, T_{\text{access}}) = T_{\text{send}} + T_{\text{recv}} + T_{\text{access}} \tag{6}$$

The total delay considering the cloudlet selection delay and offloading delay (calculated using Eq. (3)) is given as:

$$T_{\text{total}} = f(T_{\text{sc}}, T_{\text{offloadclt}}) = T_{\text{sc}} + T_{\text{offloadclt}} \tag{7}$$

4.2 Total power consumption

The power consumption in our scheme is estimated based on:

Table 2 Format of CIT containing nearby cloudlet IDs with application type

Cloudlet ID	Executable applications
Cloudlet 1	Algebra
	Numerical
	Operational research
	Graph theory
Cloudlet 2	Game
	Web service
……..	……..
	……..
	……..
Cloudlet *N*	File creation
	Sorting
	Searching
	File conversion

(i) Power in cloudlet selection (P_{sc}),
(ii) Offloading power (P_{offload}).

The offloading power is calculated using Eq. (2). The power consumption for selecting the appropriate cloudlet considers the following:

(i) Power consumption in sending message to the nearby cloudlets (P_{send}),
(ii) Power consumption in receiving response from the nearby cloudlets (P_{recv}),
(iii) Power consumption for accessing CIT (P_{access}).

The power consumption for accessing CIT in memory to find out the appropriate cloudlet is given as:

$$P_{\text{access}} = T_{\text{access}} P_{\text{acm}} \tag{8}$$

where P_{acm} is the power consumption in accessing memory per unit time by the mobile device and T_{access} is the CIT access delay.

Therefore the power consumption in cloudlet selection is given as:

$$P_{\text{sc}} = f(P_{\text{send}}, P_{\text{recv}}, P_{\text{access}}) = P_{\text{send}} + P_{\text{recv}} + P_{\text{access}} \tag{9}$$

The total power consumption in the proposed scheme is therefore given by:

$$P_{\text{tot}} = f(P_{\text{sc}}, P_{\text{offload}}) = P_{\text{sc}} + P_{\text{offload}} \tag{10}$$

In the next section, the power and delay consumption in different types of application offloading using proposed algorithm are determined.

5 Performance evaluation

5.1 Experimental results obtained from cloudlets

The performance of the proposed strategy is evaluated using the cloudlets of testbed present in the Mobile Cloud Computing laboratory of West Bengal University of Technology (WBUT). The cloudlet-based offloading environment created in the laboratory is pictorially presented in Fig. 5.

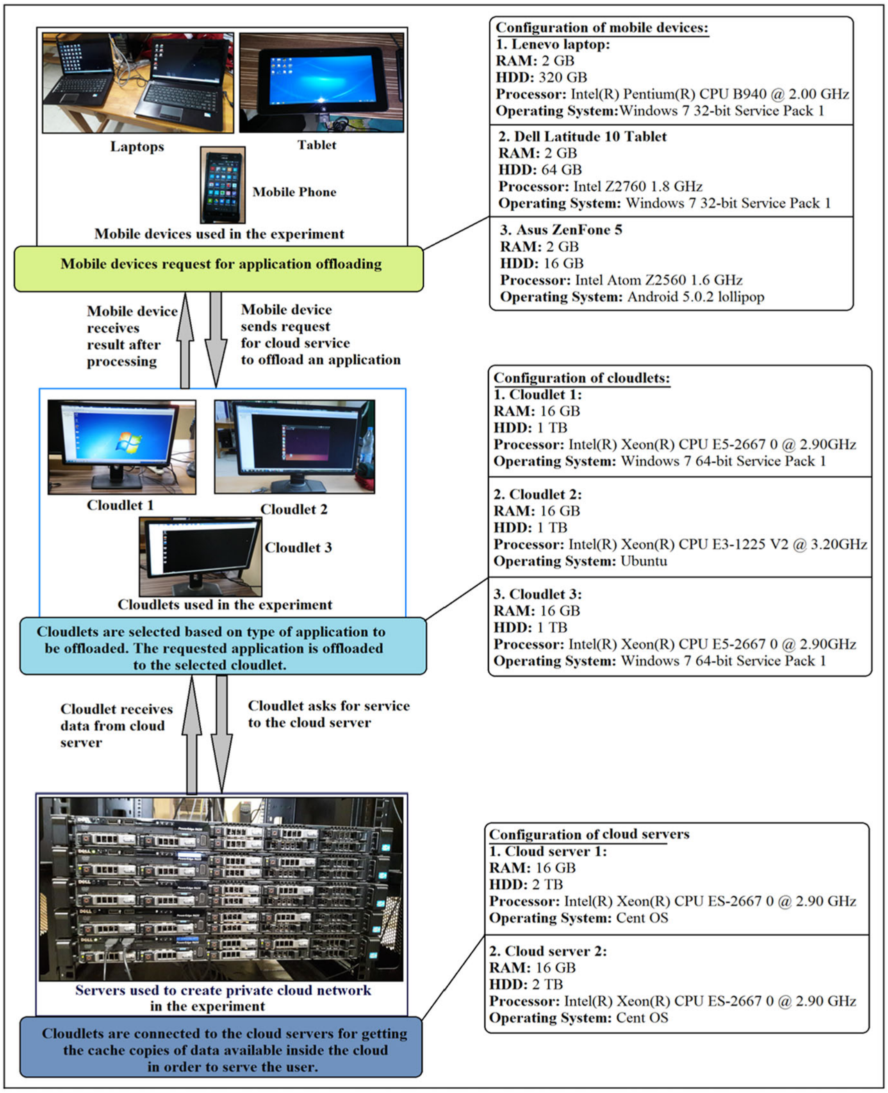

Laptops
Tablet
Mobile Phone
Mobile devices used in the experiment
Mobile devices request for application offloading
Configuration of mobile devices:
1. Lenevo laptop:
RAM: 2 GB
HDD: 320 GB
Processor: Intel(R) Pentium(R) CPU B940 @ 2.00 GHz
Operating System: Windows 7 32-bit Service Pack 1
2. Dell Latitude 10 Tablet
RAM: 2 GB
HDD: 64 GB
Processor: Intel Z2760 1.8 GHz
Operating System: Windows 7 32-bit Service Pack 1
3. Asus ZenFone 5
RAM: 2 GB
HDD: 16 GB
Processor: Intel Atom Z2560 1.6 GHz
Operating System: Android 5.0.2 lollipop
Mobile device receives result after processing
Mobile device sends request for cloud service to offload an application
Cloudlet 1
Cloudlet 2
Cloudlet 3
Cloudlets used in the experiment
Cloudlets are selected based on type of application to be offloaded. The requested application is offloaded to the selected cloudlet.
Configuration of cloudlets:
1. Cloudlet 1:
RAM: 16 GB
HDD: 1 TB
Processor: Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz
Operating System: Windows 7 64-bit Service Pack 1
2. Cloudlet 2:
RAM: 16 GB
HDD: 1 TB
Processor: Intel(R) Xeon(R) CPU E3-1225 V2 @ 3.20GHz
Operating System: Ubuntu
3. Cloudlet 3:
RAM: 16 GB
HDD: 1 TB
Processor: Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz
Operating System: Windows 7 64-bit Service Pack 1
Cloudlet receives data from cloud server
Cloudlet asks for service to the cloud server
Servers used to create private cloud network in the experiment
Cloudlets are connected to the cloud servers for getting the cache copies of data available inside the cloud in order to serve the user.
Configuration of cloud servers
1. Cloud server 1:
RAM: 16 GB
HDD: 2 TB
Processor: Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz
Operating System: Cent OS
2. Cloud server 2:
RAM: 16 GB
HDD: 2 TB
Processor: Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz
Operating System: Cent OS

Fig. 5 Experimental scenario created for performance analysis

Table 3 Configurations of cloudlets, cloud servers and mobile devices used in experiment

Sl. no.	RAM	HDD	Processor	Operating system
Cloudlet 1	16 GB	1 TB	Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90 GHz	Windows 7 64-bit service pack 1
Cloudlet 2	16 GB	1 TB	Intel(R) Xeon(R) CPU E3-1225 V2 @ 3.20 GHz	Ubuntu
Cloudlet 3	16 GB	1 TB	Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90 GHz	Windows 7 64-bit service pack 1
Cloud server 1	16 GB	2 TB	Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz	CentOS
Cloud server 2	16 GB	2 TB	Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz	CentOS
Lenevo laptop	2 GB	320 GB	Intel(R) Pentium(R) CPU B940 @ 2.00 GHz	Windows 7 32-bit service pack 1
Dell latitude 10 Tablet	2 GB	64 GB	Intel Z2760 1.8 GHz	Windows 7 32-bit service pack 1
Asus ZenFone 5	2 GB	16 GB	Intel Atom Z2560 1.6 GHz	Android 5.0.2 lollipop

Three cloudlets and two cloud servers are used in the experiment. The configurations of the cloudlets, cloud servers and mobile devices are presented in Table 3.

The six types of applications considered in our experiment are:

(A) Numerical operation
(B) Game
(C) Web service
(D) Sorting
(E) Searching
(F) File operation.

In the experiment three cloudlets are used: cloudlet 1, cloudlet 2 and cloudlet 3. Table 4 shows the delays and powers involved in offloading these application-related tasks.

The delay and power consumptions are calculated using the proposed mathematical model presented in Sect. 4. The processing delay, propagation delay and communication delay in both uplink and downlink are considered to determine the offloading delay. The cloudlet selection delay and offloading delay both are determined and the total delay is calculated and presented in Table 4. The power consumption considering the offloading and cloudlet selection is determined and presented in Table 4.

As observed from Table 4, for numerical code offloading cloudlet 1 is used. Gauss elimination, fourth-order R-K method and Euler's method are considered as three cases in numerical problem. The 4-Queens puzzle is considered as a game for which cloudlet 2 is the most suitable one according to Table 4. Cloudlet 2 is appropriate for web service oriented application also as shown in Table 4. For sorting, searching and file creation purpose cloudlet 3 is used. Quick sort and bubble sort are considered as two cases in sorting problem. Binary search and linear search are considered as

Table 4 Delay and power consumption in application offloading using cloudlets in our experiment

Cloudlet used	Application type	Offloaded process code	Deadline (s)	Delay (s)	Power (W)
Cloudlet 1	Numerical operation	Fourth-order R-K method	10	8.923	0.45
		Euler's method	10	8.564	0.43
		Gauss elimination	20	15.818	0.8
Cloudlet 2	Game	4-Queens puzzle	5	2.658	0.12
	Web service	Web service	1	0.485	0.025
Cloudlet 3	Sorting	Bubble sort	15	10.623	0.51
		Quick sort	10	7.222	0.36
	Searching	Binary search	10	8.876	0.45
		Linear search	10	9.75	0.5
	File operation	Text file creation	25	24.398	1.25
		Word file creation	30	28.501	1.45

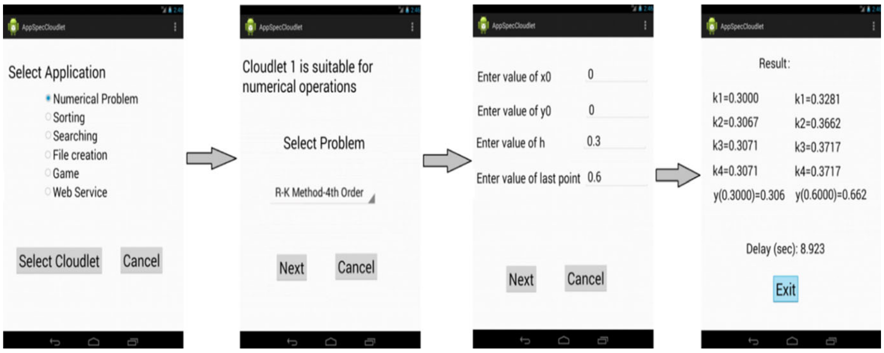
searching cases. Word and text file creations are considered in file creation task. If a request comes for a task related to numerical problem, then cloudlet 1 will be used for offloading. Else if a user requests for a game or web service oriented application, then cloudlet 2 will be selected for offloading. Else if the user request is related to the task of sorting, searching or file creation, cloudlet 3 will be selected for offloading.

In the proposed approach, secure socket programming is used. The user gives the file path as input through the user interface. After that the user gives the address of the cloudlet where the code will be offloaded depending on the application type. Then a prompt will appear asking for the password of the cloudlet. If the provided password is successfully matched with the original one, then the code is offloaded to that application-wise selected cloudlet. The total time consumption displayed in the table is counted from the time of giving file path by the user to the time of getting the result after code execution inside the cloudlet.

5.2 AppSpecCloudlet: android application for application-aware cloudlet selection

We have developed an android application "AppSpecCloudlet" in the MCC laboratory of WBUT using Android 5.0.2 lollipop. Two cases are presented where depending on the type of the requested application the appropriate cloudlet is selected. In Fig. 6 the user selects "Numerical Problem" as he or she wishes to offload the task of R-K method-4th order. It is observed from Table 5 that cloudlet 1 is suitable for offloading the numerical computation-related tasks. Hence AppSpecCloudlet selects cloudlet 1 for offloading the task of R-K method. The user enters the input values and gets solutions. As observed from Fig. 6 the delay involved in offloading this task is 8.923 s.

In Fig. 7 the user wishes to play the 4-Queens Puzzle game. According to the results of Table 5, cloudlet 2 is suitable for game offloading, thus AppSpecCloudlet selects cloudlet 2. As observed from Fig. 7 the offloading delay in this task is 2.658 s.



Cloudlet 1 is suitable for numerical operations

Enter value of x0	0
Enter value of y0	0
Enter value of h	0.3
Enter value of last point	0.6

Result:

k1=0.3000	k1=0.3281
k2=0.3067	k2=0.3662
k3=0.3071	k3=0.3717
k4=0.3071	k4=0.3717
y(0.3000)=0.306	y(0.6000)=0.662

Delay (sec): 8.923

Fig. 6 Numerical problem-related task offloading using AppSpecCloudlet

It is observed from Figs. 6 and 7 that using AppSpecCloudlet the most appropriate cloudlet for offloading a task can be selected from the perspective of the application type.

5.3 Comparison between our proposed strategy and existing methods on cloudlet

Table 5 presents the power consumption and delay in offloading different types of codes of different applications to cloudlets using existing approaches and our proposed approach. Table 5 demonstrates that the proposed scheme reduces the delay by approximately 3–27 and 2–15 % than the Round Robin algorithm [28] based approach and proxy server-based cloudlet selection scheme for offloading [29], respectively. It is also observed from Table 5 that using our proposed application-specific cloudlet selection method, the power consumption is reduced by approximately 3–35 and 2–20 %, respectively than the Round Robin algorithm [28] based approach and proxy server-based cloudlet selection algorithm for offloading [29].

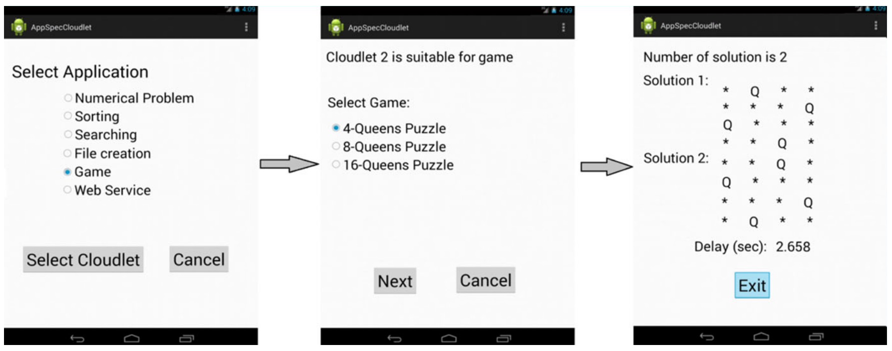
In our previous work as the offloading occurs through a proxy server, additional delay and power consumption occur. In case of Round Robin approach pre-emptive scheduling occurs and jobs are allocated in first come first serve basis. The system performance depends on the time period for pre-emptive scheduling. But in our scheme the tasks are distributed among the cloudlets based on type of application. Hence the load is distributed. This in turn reduces the delay. Consequently, the power consumption is reduced.

Table 6 presents the novelty of the proposed approach compared to the existing approaches on cloudlet. As observed from Table 6, the advantage of using cloudlet with respect to the cloud is highlighted in the existing strategies.

In most of the schemes [12,13,15,17], the selection of suitable cloudlet in a multi-cloudlet scenario has not been focused. If multiple cloudlets are available, then the user selects the nearest cloudlet in the existing scheme [13]. But if multiple cloudlets are at approximately same distance from the mobile device, then the selection is vital. The mobile device has to select the most suitable cloudlet for offloading an application. In

Table 5 Comparison of delay and power consumption in offloading application to cloudlets using our proposed scheme and existing schemes

Application type	Offloaded process code	Delay (s)			Power (W)		
		Our proposed approach	Round robin [28]	Proxy server based [29]	Our proposed approach	Round robin [28]	Proxy server based [29]
Numerical operation	Fourth-order R-K method	8.923	9.927	9.223	0.45	0.51	0.47
	Gauss elimination	15.818	16.815	16.314	0.8	0.85	0.82
Game	4-Queens puzzle	2.658	3.642	3.121	0.12	0.186	0.15
Sorting	Quick sort	7.222	7.419	7.329	0.36	0.371	0.367
Searching	Binary search	8.876	9.713	9.212	0.45	0.49	0.46
File operation	Word file creation	28.501	29.421	29.101	1.45	1.51	1.49



Cloudlet 2 is suitable for game

```
Number of solution is 2
Solution 1:  *  Q  *  *
             *  *  *  Q
             Q  *  *  *
             *  *  Q  *
Solution 2:  *  *  Q  *
             Q  *  *  *
             *  *  *  Q
             *  Q  *  *
Delay (sec): 2.658
```

Fig. 7 Game offloading using AppSpecCloudlet

our previous work the nearest cloudlet is selected as the proxy server [29]. If the proxy server is able, then it executes the code. Otherwise the proxy server offloads the task to another cloudlet selected based on low latency and low power. As the offloading occurs through the proxy server, an additional delay and power consumptions are involved. This is a shortcoming of the scheme [29]. Another issue is that different cloudlets can have different application processing abilities. In such a scenario, arbitrary selection of cloudlet can affect the QoS in terms of delay. For example, the user wants to play a game but selects a cloudlet with the ability of processing numerical applications. Then the cloudlet refuses the user request and the user has to offload the game to the cloud. But interaction with cloud increases the delay. If cloudlet selection is performed according to the type of requested application, this problem can be solved. Moreover, distributing applications throughout the cloudlets, the load on each cloudlet can be reduced at a time. When a single cloudlet with various application processing abilities exists, and different users access different types of application, the cloudlet becomes overloaded. In such a case the proper load distribution does not occur. Hence quality of user experience degrades. To solve out this issue, in our paper we have proposed application-aware cloudlet selection strategy which is a novel approach as compared to the existing cloudlet-based schemes.

6 Conclusion

The selection of cloudlet in a multi-cloudlet scenario is a promising research area. In this paper, we have proposed an application-aware cloudlet selection algorithm for multi-cloudlet environment. The proposed algorithm is implemented and its performance is analyzed using testbed located at the university laboratory. Six types of applications considered in the experimental analysis are: numerical problem, sorting and searching, file creation, game and web service. The tasks related to these applications are executed in the cloudlets and the results are delivered to the mobile devices. The delays in offloading the tasks are collected from the experiments and presented in this paper. The power consumption and delay in the proposed approach are estimated. Experimental results illustrate that our strategy reduces the delay by approximately

Table 6 Comparison between proposed method and existing cloudlet-based schemes [12,13,15,17,29]

Properties	Existing methods on cloudlet					Our proposed strategy
	VM-based cloudlet [12]	Large-scale cloudlet [13]	Augmented reality using cloudlet [15]	Resource allocation in collaborative mobile cloudlets [17]	Proxy server-based cloudlet selection for offloading in multi-cloudlet scenario [29]	
Contributions	Cloudlet-based offloading is proposed. Using cloudlet, the power consumption and delay are reduced than the cloud	Large numbers of cloudlets are allocated and mobile devices use the service provided by its nearest cloudlet. Using this approach, the power consumption and delay are reduced than the cloud	A cloudlet platform is used in augmented reality. Here the users can share their resources between each other	Optimal resource allocation is performed using two heuristic algorithms	The application is offloaded to the cloudlet selected by the proxy server based on low latency and low power	For multi-cloudlet environment application-aware cloudlet selection method is proposed
Multi-cloudlet scenario is considered	×	×	×	×	✓	✓
Most suitable cloudlet is selected among multiple cloudlets based on application type	×	×	×	×	×	✓
Remarks	In the proposed approach, application-aware cloudlet selection is performed in a multi-cloudlet scenario which is a novel strategy comparing to the existing cloudlet-based approaches					

3–27 % and the power by approximately 3–35 % than the round robin algorithm-based approach. Hence we can recommend our proposed approach for problems requiring low power and low delay as well as balancing the load of the system. In future, we wish to extend the proposed work to fog computing scenario.

Acknowledgements Department of Science and Technology (DST) for sanctioning a research Project entitled "Dynamic Optimization of Green Mobile Networks: Algorithm, Architecture and Applications" under Fast Track Young Scientist scheme reference no.: SERB/F/5044/2012-2013, DST FIST SR/FST/ETI-296/2011 and No.DST/INSPIRE Fellowship/2013/327,TEQIP-II under which this work has been completed.

References

1. Buyya R, Broberg J, Goscinski AM (2010) Cloud computing: principles and paradigms. Wiley, New York
2. Lu G, Zeng WH (2014) Cloud computing survey. Appl Mech Mater 530:650–661
3. Buyya R, Beloglazov A, Abawajy J (2010) Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010), July 12–15 2010, CSREA Press, Las Vegas
4. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Future Gener Comput Syst 29:84–106
5. Dinh HT, Lee C, Niyato D, Wang P (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mob Comput 13:1587–1611
6. Mukherjee A, De D (2016) Low power offloading strategy for femto-cloud mobile network. Eng Sci Technol Int J 19:260–270
7. Mukherjee A, Gupta P, De D (2014) Mobile cloud computing based energy efficient offloading strategies for femtocell network. In: Applications and innovations in mobile computing, IEEE, pp 28–35
8. Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R (2014) Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. Commun Surv Tutor IEEE 16:337–368
9. Ahmed E, Gani A, Khan MK, Buyya R, Khan SU (2015) Seamless application execution in mobile cloud computing: motivation, taxonomy, and open challenges. J Netw Comput Appl 52:154–172
10. Ahmed E, Akhunzada A, Whaiduzzaman M, Gani A, Ab Hamid SH, Buyya R (2015) Network-centric performance analysis of runtime application migration in mobile cloud computing. Simul Model Pract Theory 50:42–56
11. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28:755–768
12. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. Pervasive Comput IEEE 8:14–23
13. Tawalbeh LA, Jararweh Y, Dosari F (2015) Large scale cloudlets deployment for efficient mobile cloud computing. J Netw 10:70–76
14. Quwaider M, Jararweh Y (2015) Cloudlet-based efficient data collection in wireless body area networks. Simul Model Pract Theory 50:57–71
15. Verbelen T, Simoens P, Turck FD, Dhoedt B (2014) Adaptive deployment and configuration for mobile augmented reality in the cloudlet. J Netw Comput Appl 41:206–216
16. Duro FR, Blas JG, Higuero D, Perez O, Carretero J (2015) CoSMiC: a hierarchical cloudlet-based storage architecture for mobile clouds. Simul Model Pract Theory 50:3–19
17. Bohez S, Verbelen T, Simoens P, Dhoedt B (2015) Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets. Simul Model Pract Theory 50:109–129
18. O'Sullivan MJ, Grigoras D (2015) Integrating mobile and cloud resources management using the cloud personal assistant. Simul Model Pract Theory 50:20–41
19. Ding D, Fan X, Luo S (2015) User-oriented cloud resource scheduling with feedback integration. J Supercomput 72:3114–3135
20. Aminzadeh N, Sanaei Z, Ab Hamid SH (2015) Mobile storage augmentation in mobile cloud computing: taxonomy, approaches, and open issues. Simul Model Pract Theory 50:96–108

21. Singh S, Chana I (2015) QRSF: QoS-aware resource scheduling framework in cloud computing. J Supercomput 71:241–292
22. Li C (2012) Optimal resource provisioning for cloud computing environment. J Supercomput 62:989–1022
23. Sood SK, Sandhu R (2015) Matrix based proactive resource provisioning in mobile cloud environment. Simul Model Pract Theory 50:83–95
24. Liu X, Li S, Tong W (2015) A queuing model considering resources sharing for cloud service performance. J Supercomput 71:4042–4055
25. Shiraz M, Ahmed E, Gani A, Han Q (2014) Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. J Supercomput 67:84–103
26. Li B, Pei Y, Wu H, Shen B (2015) Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds. J Supercomput 71:3009–3036
27. Shiraz M, Gani A (2014) A lightweight active service migration framework for computational offloading in mobile cloud computing. J Supercomput 68:978–995
28. Samal P, Mishra P (2013) Analysis of variants in round robin algorithms for load balancing in cloud computing. Int J Comput Sci Inf Technol 4:416–419
29. Mukherjee A, De D, Roy D G (2016) A power And latency aware cloudlet selection strategy for multi-cloudlet environment. IEEE Trans Cloud Comput 1