A Power and Latency Aware Cloudlet Selection Strategy for Multi-Cloudlet Environment

Anwesha Mukherjee, *Student Member*, IEEE, Debashis De, *Senior Member*, IEEE, and Deepsubhra Guha Roy, *Student Member*, IEEE

Abstract—Fast interactive response in mobile cloud computing is an emerging area of interest. Execution of applications inside the remote cloud increases the delay and affects the service quality. To avoid this difficulty cloudlet is introduced. Cloudlet provides the same service to the device as cloud at low latency but at high bandwidth. But selection of a cloudlet for offloading computation at low power is a major challenge if more than one cloudlet is available nearby. In this paper we have proposed a power and latency aware optimum cloudlet selection strategy for multi-cloudlet environment with the introduction of a proxy server. Theoretical analysis show that using the proposed approach the power and the latency consumption are reduced by approximately 29-32% and 33-36% respectively than offloading to the remote cloud. An experimental analysis of the proposed cloudlet selection scheme is performed using cloudlets and cloud servers located at our university laboratory. Theoretical and experimental results demonstrate that using the proposed strategy power and latency aware cloudlet selection can be performed. The proposed approach is compared with the existing methods on multi-cloudlet scenario to demonstrate that the proposed approach reduces the power consumption and the system response time.

Index Terms— Multi-cloudlet environment, optimum cloudlet, proxy server, latency, power, system response time.

1 Introduction

THE number of mobile subscribers has increased explosively in the last few decades due to the advancement in mobile network and technologies. With this advancement mobile cloud computing [1] has become an emerging area of research. With the help of mobile cloud computing most of the application processing is left to the cloud and the result is sent back to the mobile device. This method is called offloading which saves the processing power of the mobile device [2]. Offloading and storage to the cloud has removed the drawbacks of limited storage and limited processing power of resource-constrained mobile devices [3]. But offloading application to the remote cloud introduces communication overhead which increases the wide area network (WAN) delay and cost. As a result the quality of service (QoS) gets affected. To solve out this issue, cloudlet is introduced. Cloudlet is a resource-rich computer or a cluster of computers which contains the soft-state like cache copies of the data already available in the cloud [4-5]. Offloading to a cloudlet provides high bandwidth, low latency and low cost wireless access to the network [6-8]. When a mobile device does not wish to offload to the cloud due to delay, cost etc, it can find a nearby cloudlet. Mobile devices like laptop, tablet and mobile phone are connected to the nearby cloudlet which is connected to the cloud as shown in Fig.1. Cloudlet based architecture is an agent-client architecture where the mobile devices are connected with the cloud through the agent cloudlet [4-5] as shown in Fig.1. When a mobile

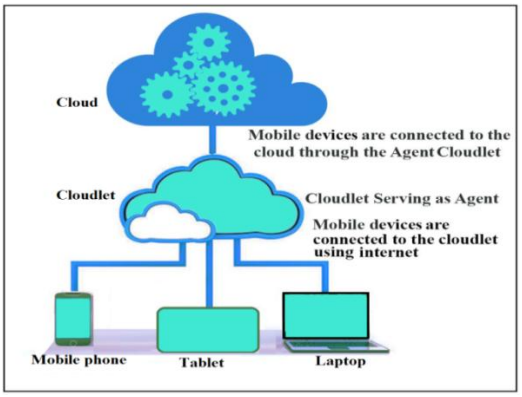


Fig.1. Mobile devices are connected to the cloud through the agent cloudlet

- *Anwesha Mukherjee is pursuing PhD in West Bengal University of Technology, BF-142, Sector-I, Salt lake, Kolkata-700064, West Bengal, India. E-mail: anweshamukherjee2011@gmail.com.*
- *Debashis De is an Associate Professor and presently Head of the Department of Computer Science & Engineering, West Bengal University of Technology, BF-142, Sector-I, Salt lake, Kolkata-700064, West Bengal, India. E-mail: dr.debashis.de@ieee.org.*
- *Deepsubhra Guha Roy is pursuing PhD in West Bengal University of Technology, BF-142, Sector-I, Salt lake, Kolkata-700064, West Bengal, India. E-mail: roysubhraguha@gmail.com.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

device asks for a service to the cloudlet, it sends a reply if it is able to serve the device. Otherwise it forwards the request to the cloud. After receiving reply from the cloud, the cloudlet sends the result to the device.

1.1 Motivations and Contributions of Proposed Work

Cloudlet based offloading is a promising research area [8]. Cloudlet provides cloud services at low latency. But selection of cloudlet is vital when multiple cloudlets are available nearby. In the existing works when multiple cloudlets exist nearby, the nearest cloudlet is selected for low latency. It may possible that the selected nearest cloudlet is unable to execute the requested computation. In that case the cloud is selected for offloading. If multiple cloudlets are available nearby the mobile device, then offloading to the cloud consumes more latency as well as more power. Our motivation is to deal with this problem. The intuition of the proposed work is to provide a low power and low latency offloading facility to the mobile users if multiple cloudlets are available. The contributions of this paper are:

1. From a resource-hungry mobile device to offload a code of an application for a given input, an optimum cloudlet selection method is proposed for multi-cloudlet environment with introduction of a proxy server.
2. The power and latency consumption models of the proposed method are developed.
3. The simulation and experimental results show that the proposed strategy is a power and latency aware offloading method.

The organization of the paper is: section 2 contains the related works, section 3 contains the proposed optimum cloudlet selection scheme for multi-cloudlet environment, the power and latency consumption models of the proposed method are developed in section 4, section 5 presents experimental results analyzing the performance of the proposed scheme and a comparative analysis with the existing schemes, and section 6 concludes the paper.

2 Related Works

Nowadays people are willing to use mobile devices like laptop, smart phone, tablet, i-pad etc rather than the immobile desktop computers. To provide high speed internet access in these hand-held mobile devices, various challenges come into the scenario like lack of storage, computational power, limited resource and limited battery life of mobile devices. To overcome these difficulties, cloud computing is incorporated into the mobile network technologies. Cloud computing (CC) offers a distributed environment which provides shared access to a collection of configurable resources in an on-demand and pay-as-you-use basis to the users [9]. Cloud is the combination of virtualization of high amount of resources with a distributed computing paradigm incorporated with Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [9-12]. For resource allocation and migration inside the cloud, various strategies are discussed in [13-16]. A scheduling strategy with resource provisioning is proposed for IaaS scenario in [15] based on dead line. Another resource management strategy is proposed for IaaS in [16]. A hyper-heuristic scheduling strategy for cloud is proposed in [17]. An energy-efficient scheduling strategy for cloud is proposed in [18] where the virtual machines (VMs) are allocated, migrated and cancelled dynamically to make the cloud scalable. Integration of mobile computing with CC has given birth of a newer technological approach mobile cloud computing (MCC) [19]. Using MCC the application processing and storage are left to the cloud instead of the mobile device. The process of executing partial or full part of the resource hungry applications to the cloud instead of performing it within the mobile device is referred as offloading [2]. MCC is related to the wide area network (WAN) and the cloud. But access to the remote cloud increases the WAN delay. To deal with this problem, cloudlet is introduced as a resource-rich computer or a cluster of computers containing the cache copies of the data available in the cloud. Cloudlet can also be used for personal data storage like cloud. In [7] a cloudlet allocation method is proposed for improving QoS of the system. For wireless body area network (WBAN) [20-21], a health data collection method is proposed in [22] where the cloudlet is used to reduce the delay and the power consumption involved in collecting data. A dynamic offloading strategy for cloudlet is proposed in [23] for reducing the energy consumption. Although various methods are proposed for health monitoring, augmented reality, and storage based on cloudlets [22, 24], the power and latency aware cloudlet selection in a multi-cloudlet environment is a promising research scope. In [25], an optimal cloudlet allocation strategy is proposed where multiple cloudlets are allocated in a densely populated area. The cloudlets are allocated to the users in such a manner that the workload is balanced and the system response time is reduced [25]. In [26], multiple VMs and cloudlets are considered where cloudlets are allocated to VMs in a way that the service quality is improved. In [27] it is demonstrated whether offloading saves energy or not. In this paper we have proposed a novel strategy for optimum cloudlet selection based on low power and low latency with the introduction of a proxy server.

3 Proposed Optimum Cloudlet Selection Strategy

In application offloading it has to be checked whether offloading is power saving or not. But same application can have different number of instructions due to conditional statements [28-29]. For different data input and different branches, the same application may have different number of instructions. Due to this reason, in this paper we have performed code offloading of an application for a given input. The user requests for offloading a particular code of an application with the input values. From a mobile device the process of offloading a code of an application with input values to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

TABLE 1
PARAMETERS USED IN POWER AND LATENCY CALCULATION

Parameter	Definition	Value
B_u	Uplink data transmission rate	50-80 mbps
B_d	Downlink data transmission rate	100-300 mbps
P_{ts}	Power consumed by mobile device for sending data per unit time	10-20 mW/sec
P_{tr}	Power consumed by mobile device for receiving data per unit time	5-15 mW/sec
U_f	Uplink failure rate	0.005-0.01
D_f	Downlink failure rate	0.15-0.2
D_{cl}	Distance of cloudlet from mobile device	10-50m
S_p	Propagation speed	3 x 10^8 m/s
D_c	Distance of cloud from mobile device	1-10km

the cloudlet is presented in Fig. 2. The system consists of three main components:

- Mobile device
- Cloudlet
- Cloud

A mobile device is connected to the cloudlet which is connected to the cloud.

The parameters used in power and latency calculation in cloudlet based offloading are given in TABLE 1 [2, 30].

If the mobile device performs the computation, the power consumed is given as [2],

$$P_M = P_{mo} \cdot (I / S_M) \tag{1}$$

where P_{mo} is the power consumed by the mobile device per unit time during computation, S_M is the speed of the mobile device denoting the number of instructions executed per unit time by it and I is the number of instructions to be executed for the particular code with given input.

If the cloudlet performs the operation, the power consumed by the mobile device during the offloading period is determined as [2],

$$\begin{aligned} P_{CL} &= P_i \cdot [(I / S_{cl}) + (D_{cl} / S_p) + T_{clq}] \\ &+ P_{ts} \cdot [(1+U_f)(D_u / B_u)] \\ &+ P_{tr} \cdot [(1+D_f)(D_d / B_d)] \end{aligned} \tag{2}$$

where P_i is the power consumed by the mobile device per unit time during offloading, S_{cl} is the speed of the cloudlet denoting the number of instructions executed per unit time by the cloudlet, D_u is the total amount of data transmitted in bits in uplink, D_d is the total amount of data transmitted in bits in downlink and T_{clq} is the queuing latency.

The power saving obtained when the code is executed by the cloudlet, is given by,

$$P_{save} = P_M - P_{CL} \tag{3}$$

If $P_{save} \le 0$, then the job is executed within the mobile device. Otherwise the job deadline is compared with the latency in case of offloading. To calculate the latency, the propagation latency, the communication latency, the processing latency and the queuing latency are considered.

The propagation latency for the cloudlet is computed as,

$$T_{clpr} = D_{cl} / S_p \tag{4}$$

Total uplink communication latency with the cloudlet is calculated as [4],

$$T_{clu} = (1+U_f)(D_u / B_u) \tag{5}$$

Total downlink communication latency with the cloudlet is computed as [4],

$$T_{cld} = (1+D_f)(D_d / B_d) \tag{6}$$

Total communication latency with the cloudlet is given as,

$$\begin{aligned} T_{clc} &= T_{clu} + T_{cld} = (1+U_f)(D_u / B_u) \\ &+ (1+D_f)(D_d / B_d) \end{aligned} \tag{7}$$

The processing latency for executing I instructions in the cloudlet is given by,

$$T_{clp} = I / S_{cl} \tag{8}$$

The total latency in offloading the code to the cloudlet considering the propagation latency, the communication latency, the processing latency and the queuing latency (T_{clq}), is calculated as,

$$\begin{aligned} T_{cl} &= T_{clpr} + T_{clc} + T_{clp} + T_{clq} \\ &= [D_{cl} / S_p] + [(1+U_f)(D_u / B_u) \\ &+ (1+D_f)(D_d / B_d)] + [I / S_{cl}] + T_{clq} \end{aligned} \tag{9}$$

This latency is compared with the job deadline. If the job deadline is less than or equals to the latency, the code is executed within the mobile device to avoid the miss of deadline. Else the code is offloaded to the cloud. In our proposed scheme we have considered the power consumption as well as latency to decide whether to offload or not. The proposed cloudlet selection algorithm is presented in TABLE 2. The working model of the proposed scheme is pictorially depicted in Fig.2.

As observed from TABLE 2, three cases are possible in the proposed strategy:

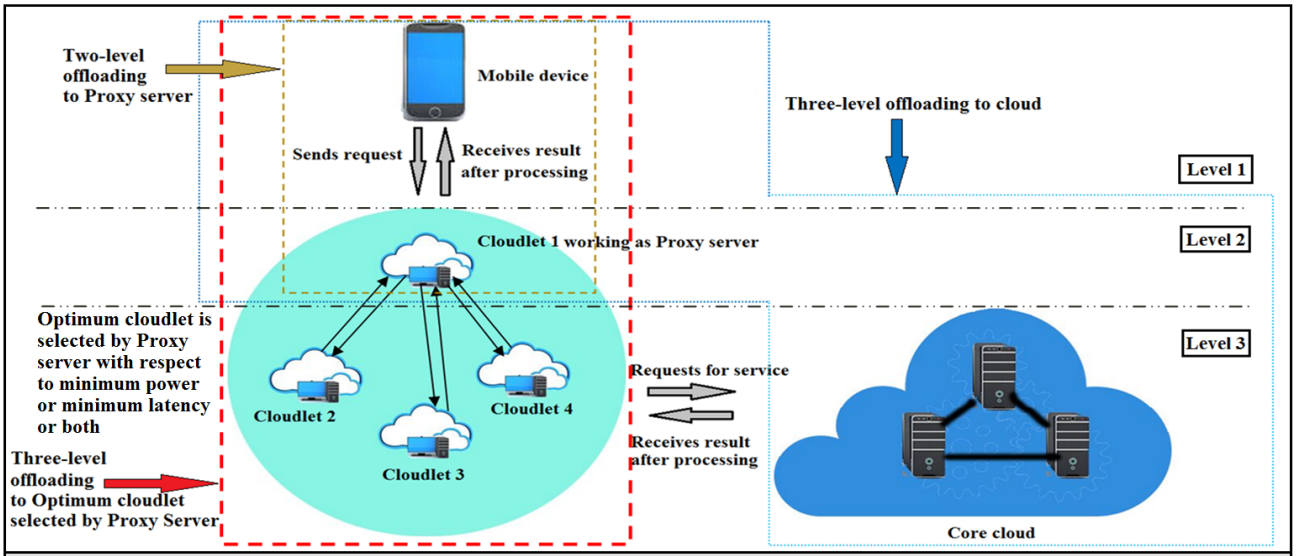
Case 1: Two-level offloading to nearest cloudlet acting as proxy server- The nearest cloudlet i.e. proxy server is able to satisfy the user's request and the code is offloaded to that cloudlet. In this case level 1 contains the mobile device and level 2 contains the nearest cloudlet i.e. proxy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

server.

Case 2: Three-level offloading to optimum cloudlet selected by proxy server- The nearest cloudlet is unable to serve the user's request. Therefore the nearest cloudlet acts as a proxy server and selects the optimum cloudlet among its nearby cloudlets with respect to minimum power consumption or minimum latency or both to offload the code. In this case level 1 contains the mobile device, level 2 contains the proxy server and level 3 contains the selected optimum cloudlet.

Case 3: Three-level offloading to cloud- If none of the nearby cloudlets can satisfy the user's request, the code is offloaded to the cloud through the proxy server. In this case level 1 contains the mobile device, level 2 contains the proxy server and level 3 contains the cloud.



Proposed Optimum Cloudlet Selection Method

1. A mobile device requests for offloading a code of an application with input to nearest cloudlet.
2. If the cloudlet is able to execute the code, it sends back the result to the mobile device after processing.
3. Else the cloudlet acts as a proxy server and searches for a nearby cloudlet by broadcasting a request message for the required service.
4. The nearby cloudlets which are able to satisfy the request, send responses to the proxy server with the information about their speed and processing power.
5. Based on these information, the proxy server calculates the power and latency consumed in executing the code by the cloudlets for the given input.
6. The proxy server selects the optimum cloudlet with minimum power and minimum latency.
7. If no such cloudlet is found, based on the priority of either minimum power or minimum latency, the optimum cloudlet is selected by the proxy server.
8. The code is offloaded to the selected optimum cloudlet.
9. After execution the result is sent back to the mobile device.
10. Else if no cloudlet is found nearby for offloading the code, it is offloaded to the cloud by the proxy server.

Fig.2. Offloading from a mobile device to the cloudlet using the proposed strategy

TABLE 2
PROPOSED OPTIMUM CLOUDLET SELECTION ALGORITHM

Considerations:

P_{Mi}: Power consumption in executing a code c_i of an application a_i inside the requested mobile device for a given input i_p.

P_{CLi}: Power consumption in offloading the code c_i of application a_i to the cloudlet for the input i_p.

P_{savei}: Power saving in offloading the code c_i of application a_i with input i_p.

T_{CLi}: Total latency in offloading the code c_i of application a_i with input i_p to the cloudlet.

Dl_i: Deadline of the code c_i of application a_i with input i_p.

T_{maxi}: Maximum waiting time of proxy server for receiving response from nearby cloudlets after broadcasting request message for execution of the code c_i of application a_i with input i_p.

C_{minP}: Cloudlet providing minimum power consumption.

C_{minT}: Cloudlet providing minimum latency.

C_{opt}: Optimum cloudlet selected by proxy server.

Algorithm:

1. **Start**

/*Decision regarding local or remote execution*/

2. **If** a mobile device user wants to offload the code c_i with input i_p,
3. It calculates the power consumption in executing the code c_i inside the mobile device (P_{Mi}) using equation (1) and the power consumption in offloading the code c_i to the cloudlet (P_{CLi}) using equation (2) for the given input i_p
4. The power saving (P_{savei}) in offloading the code c_i with input i_p is calculated using equation (3)
5. **If** $P_{savei} \leq 0$, /* Code offloading does not save power */
6. Execute the code c_i with input i_p locally within the mobile device
7. **Else** /*Code offloading saves power*/
8. The total latency (T_{CLi}) in offloading the code c_i with input i_p to the cloudlet is calculated using equation (9)
9. **If** $Dl_i \leq T_{CLi}$, /*Deadline of code c_i is less than or equals to the total latency in offloading c_i to cloudlet*/
10. The code c_i with input i_p is locally executed inside the mobile device
11. **Else** /*Deadline of code c_i is greater than the total latency in offloading c_i to cloudlet*/
12. The mobile device searches for a nearby cloudlet using Service Location Protocol [31] for offloading the code c_i with input i_p

/*Selection of optimum cloudlet*/

13. **If** any cloudlet is found nearby,
14. The nearest cloudlet is selected to receive the code c_i with input i_p

/* Two-level offloading to nearest cloudlet acting as proxy server*/

15. **If** the nearest cloudlet is able to execute the requested code c_i with input i_p,
16. It executes the code c_i with input i_p and sends back the result to the requested mobile device

/* Three-level offloading to optimum cloudlet selected by proxy server*/

17. **Else if** the nearest cloudlet is unable to execute the code c_i with input i_p,
18. It acts as a proxy server and searches for a nearby cloudlet by broadcasting a request message for the required service and sets a timer t to T_{maxi} where $T_{\max i} << T_{CLi}$
19. **If** the nearby cloudlets are able to satisfy the request,
20. They send responses to the proxy server with the information regarding their speed and processing power
21. The proxy server calculates and compares the power consumptions and the latencies involved in executing the code c_i with input i_p by the nearby cloudlets whose responses are received before the timer expires, based on the code size and type, the distances of the cloudlets from the proxy server, the speed and the processing power of the cloudlets
22. The proxy server selects the optimum cloudlet (C_{opt}) providing minimum power and minimum latency using equation (31) as $(C_{\min P} \cap C_{\min T})$
23. **If** no cloudlet with minimum power consumption as well as minimum latency is found,
24. **If** the priority is minimum power consumption, the cloudlet (C_{minP}) providing minimum power consumption is selected by the proxy server using equation (27) and referred as optimum cloudlet $C_{opt} \leftarrow C_{\min P}$
25. **Else if** the priority is minimum latency, the cloudlet (C_{minT}) providing minimum latency is selected by the proxy server using equation (29) and referred as optimum cloudlet $C_{opt} \leftarrow C_{\min T}$
26. **End if**
27. **End if**
28. The code c_i with input i_p is offloaded to the selected optimum cloudlet
29. **End if**
30. **End if**

/* Three-level offloading to cloud*/

31. **Else if** no cloudlet is found nearby to offload the code c_i with input i_p and the timer value reaches to 0,
32. The code c_i with input i_p is offloaded to the cloud
33. **End if**
34. **End if**
35. **End if**
36. **End if**
37. **End**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

3.1 Time Complexity of Proposed Algorithm

In the proposed strategy the proxy server either executes the requested code with the given input if it is able or broadcasts a request message to the nearby cloudlets if it is unable to process it. The nearby cloudlets send responses to the proxy server along with the information regarding their speed and processing power. Based on these information the optimum cloudlet offering low power and low latency is selected for offloading that code with the given input. If the proxy server is able to execute the code for the given input, the time involved is calculated using equation (9). Otherwise the selection time is determined. The sum of the selection time and the offloading time is the total time in this case. If the time required between sending message and receiving response for a single cloudlet C_j is T_j, and there are N number of cloudlets available nearby, the message receiving time is given by,

$$T_m(N) = \max(T_1, T_2, ...T_N) \tag{10}$$

If there are N cloudlets, the search time required to find out the cloudlet with lowest latency or lowest power consumption is given as,

$$T_s(N) = N * t_s \tag{11}$$

where t_s is time required in comparing the latency or the power consumption of two cloudlets. If the cloudlet with lowest latency gives lowest power consumption in executing the code, it is selected. Else depending on the user priority of low latency or low power consumption, the optimum cloudlet is chosen. Hence the total search time to find out the optimum cloudlet providing minimum power and minimum latency is given as,

$$T_{ser}(N) = T_s(N) + T_s(N) = 2 * N * t_s \tag{12}$$

The optimum cloudlet selection time is determined as the sum of the message receiving time and the total search time, given as,

$$T_{opsel}(N) = T_m(N) + T_{ser}(N) = T_m(N) + 2 * N * t_s \tag{13}$$

As the distances between the proxy server and nearby cloudlets are very less, the message receiving time is very small with respect to the total search time, i.e. $T_m(N) < T_{ser}(N)$. Therefore the time complexity of the proposed optimum cloudlet selection algorithm for offloading the code is given as,

$$T_{opsel}(N) = O(N) \tag{14}$$

The total time or latency for offloading the code using the proposed algorithm is given as,

$$T_{off} = T_{opsel}(N) + T_{cl} \tag{15}$$

where T_{cl} is the offloading latency to the cloudlet. If the proxy server executes the requested code, then $T_{opsel}(N) = 0$. If none of the nearby cloudlets responds, then cloud is used to offload the code. In that case the proxy server waits for T_{maxi} time as per the proposed algorithm, therefore $T_{opsel}(N) = T_{\max i}$. If there are A number of requests arrive for code offloading and the offloading time of code c_i with given input i_p is T_{offi}, the system response time is calculated as [27],

$$T_{res} = \frac{1}{A} \sum_{c_i} T_{offi} \tag{16}$$

The system response time in our approach is determined and compared with the existing schemes in section 5.3.

4 Power and Latency Consumption Model of Proposed Strategy

4.1 Power Consumption of Proposed Strategy

If case 1 occurs, the power consumed is determined as,

$$P_{case1} = P_i \cdot [(I / S_{ps}) + (D_{ps} / S_p) + T_{psq}] + P_{ts} \cdot [(1 + U_f)(D_u / B_u)] + P_{tr} \cdot [(1 + D_f)(D_d / B_d)] \tag{17}$$

where S_{ps} is the speed of the proxy server, D_{ps} is the distance of the proxy server from the mobile device and T_{psq} is the queuing latency in case of the proxy server.

If case 2 occurs, the power consumed is determined as,

$$P_{case2} = P_i \cdot [(I / S_{cls}) + (D_{ps} / S_p) + (D_{cls} / S_p) + T_{clsq}] + P_{ts} \cdot [(1 + U_f)(D_u / B_u)] + P_{tr} \cdot [(1 + D_f)(D_d / B_d)] + (P_i \cdot T_{opsel}(N)) \tag{18}$$

where S_{cls} is the speed of the selected optimum cloudlet, D_{ps} is the distance of the proxy server from the mobile device, D_{cls} is the distance of the selected optimum cloudlet from the proxy server, T_{clsq} is the queuing latency in case of the selected optimum cloudlet and $T_{opsel}(N)$ is the optimum cloudlet selection time determined using equation (13).

If case 3 occurs, the power consumed is determined as,

$$P_{case3} = P_i \cdot [(I / S_c) + (D_{ps} / S_p) + (D_c / S_p) + T_{cq}] + P_{ts} \cdot [(1 + U_f)(D_u / B_u)] + P_{tr} \cdot [(1 + D_f)(D_d / B_d)] + (P_i \cdot T_{\max i}) \tag{19}$$

where S_c is the speed of the cloud, D_{ps} is the distance of the proxy server from the mobile device, D_c is the distance of the cloud from the proxy server, T_{cq} is the queuing latency in case of the cloud and T_{maxi} is the waiting time of the proxy server.

Let the probability of occurrence of case 1, 2 and 3 are p_1, p_2, and p_3 respectively. Then the power consumption in the proposed offloading scheme is calculated as,

$$P_{pro} = p_1 \cdot P_{case1} + p_2 \cdot P_{case2} + p_3 \cdot P_{case3} \tag{20}$$

where $p_1 \le 1$, $p_2 \le 1$, $p_3 \le 1$ and $p_1 + p_2 + p_3 = 1$.

4.2 Latency Consumption of Proposed Strategy

If case 1 occurs, the total latency consumed is given by,

$$T_{case1} = [D_{ps} / S_p] + [(1 + U_f)(D_u / B_u) + (1 + D_f)(D_d / B_d)] + [I / S_{ps}] + T_{psq} \tag{21}$$

where the propagation latency is $T_{clpr1} = (D_{ps} / S_p)$.

In case 2, the propagation latency is determined as,

$$T_{clpr2} = (D_{ps} / S_p) + (D_{cls} / S_p) \tag{22}$$

where D_{ps} is the distance of the proxy server from the mobile device and D_{cls} is the distance of the selected optimum cloudlet from the proxy server. In case 2, the optimum cloudlet is first selected and the code is offloaded to that cloudlet.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

Therefore the total latency in case 2 is given by,

$$T_{case2} = T_{opsel}(N) + T_{clpr2} + [(1+U_f)(D_u / B_u) + (1+D_f)(D_d / B_d)] + [I / S_{cls}] + T_{clsq} \quad (23)$$

If case 3 occurs, the propagation latency is calculated as,

$$T_{clpr3} = (D_{ps} / S_p) + (D_c / S_p) \quad (24)$$

where D_{ps} is the distance of the proxy server from the mobile device and D_c is the distance of the cloud from the proxy server. Hence the total latency in case 3 is given by,

$$T_{case3} = T_{\max i} + T_{clpr3} + [(1+U_f)(D_u / B_u) + (1+D_f)(D_d / B_d)] + [I / S_c] + T_{cq} \quad (25)$$

Thus the latency consumed in the proposed approach is calculated as,

$$T_{pro} = p_1 \cdot T_{case1} + p_2 \cdot T_{case2} + p_3 \cdot T_{case3} \quad (26)$$

where $p_1 \le 1$, $p_2 \le 1$, $p_3 \le 1$ and $p_1 + p_2 + p_3 = 1$.

4.3 Selection of Cloudlet with Minimum Power Consumption

Let there be N cloudlets C_1, C_2 ...C_N available nearby and the power consumption in offloading the requested code with given input to these cloudlets are P_1, P_2..... P_N respectively. The cloudlet providing minimum power consumption is selected as,

$$C_{\min P} = \{C_K\} \quad (27)$$

where $C_K \in \{C_1, C_2, ..., C_N\}$ and the power consumption using C_K is P_K given as,

$$P_K = \min(P_1, P_2, ..., P_N) \quad (28)$$

4.4 Selection of Cloudlet with Minimum Latency

Let there be N cloudlets C_1, C_2 ...C_N available nearby and the latency consumption in offloading the requested code with given input to these cloudlets are T_1, T_2..... T_N respectively. The cloudlet providing minimum latency is selected as,

$$C_{\min T} = \{C_J\} \quad (29)$$

where $C_J \in \{C_1, C_2, ..., C_N\}$ and the latency using C_J is T_J given as,

$$T_J = \min(T_1, T_2, ..., T_N) \quad (30)$$

4.5 Selection of Optimum Cloudlet with Minimum Power Consumption and Minimum Latency

The optimum cloudlet providing minimum power consumption and minimum latency is selected as,

$$C_{opt} = C_{\min P} \cap C_{\min T} \quad (31)$$

The code is offloaded to the selected optimum cloudlet. If no such cloudlet satisfying both the conditions is found nearby, depending on the priority of low power consumption or low latency, the optimum cloudlet is selected by the proxy server for offloading that code with the given input.

In Fig.3 and Fig.4 the power consumption and the latency in the proposed offloading scheme are pictorially compared with cloud based offloading [2]. The power consumption is measured in watt (W) and the latency is measured in second (sec). Fig.3 presents the power consumption in the proposed scheme calculated using equation (20) and in the cloud based offloading. From Fig.3 it is observed that the power consumed in the proposed cloudlet based scheme is approximately <2 W whereas if the cloud based offloading [2] is used the power consumption is approximately <3 W. Theoretical results present that using the proposed scheme the power consumption can be reduced by approximately 29-32% than the remote cloud based offloading. Fig.4 presents the latency consumed in the proposed scheme calculated using equation (26) and in the cloud based offloading. From Fig.4 it is demonstrated that the latency in the proposed cloudlet based offloading scheme is approximately 20-50 sec whereas in the cloud based offloading the latency is approximately 30-70 sec. It is observed from the theoretical results that using the proposed scheme the latency can be reduced by approximately 33-36% than the remote cloud based offloading. Hence the proposed approach can be referred as a power and latency aware approach. In section 5.3 the proposed scheme is compared with the existing schemes with respect to power consumption and system response time.

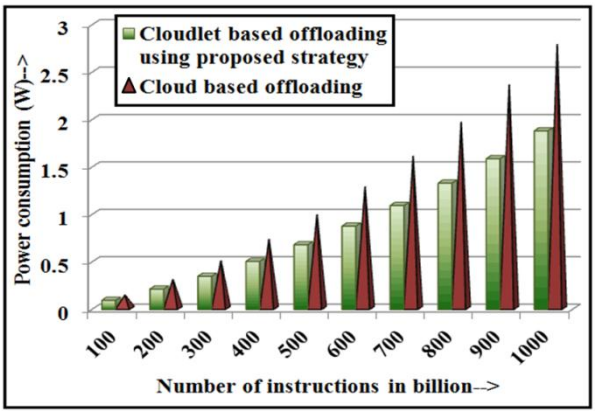


Fig.3. Power consumed in cloudlet based offloading using proposed strategy and in cloud based offloading

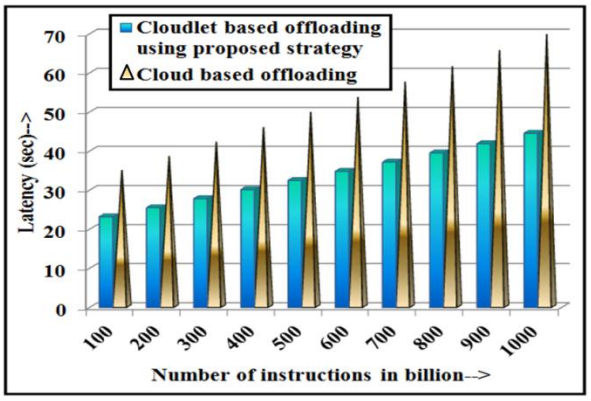


Fig.4. Latency consumed in cloudlet based offloading using proposed strategy and in cloud based offloading

TABLE 3
CONFIGURATIONS OF CLOUDLETS AND CLOUD SERVERS USED IN EXPERIMENT

Sl. no.	RAM	HDD	Processor	Operating System
Cloudlet 1	16GB	1TB	Intel(R) Xeon(R) CPU E3-1225 V2 @ 3.20GHz (Quad Core)	Ubuntu
Cloudlet 2	16GB	1TB	Intel(R) Xeon(R) CPU E3-1225 V2 @ 3.20GHz (Quad Core)	Ubuntu
Cloudlet 3	16GB	1TB	Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz (Octa Core)	Windows 7 64-bit Service Pack 1
Cloudlet 4	16GB	1TB	Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz (Octa Core)	Windows 7 64-bit Service Pack 1
Cloud Server 1	16GB	2TB	Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz (Hexa Core)	CentOS
Cloud Server 2	16GB	2TB	Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz (Hexa Core)	CentOS

5 PERFORMANCE ANALYSIS OF PROPOSED STRATEGY

5.1 Experimental Analysis of Proposed Strategy using Cloudlets located at University Campus

The proposed strategy for selecting optimum cloudlet is analyzed based on the experimental results obtained using the cloudlets located at the campus of West Bengal University of Technology (WBUT). We have four cloudlets in the MCC laboratory of our university. The configurations of these cloudlets and the cloud servers used in our experiment are presented in TABLE 3. The mobile device used in this experiment is Samsung smart phone, Asus ZenFone 5 with 2 GB RAM, 16 GB storage and Intel Atom Z2560 1.6 GHz processor. This device requests for execution of the following processes:

i) Code of file creation,
ii) Code of matrix multiplication,
iii) Code of N-Queens puzzle,
iv) Code of character array sorting.

The cloudlet 1 is nearest to the mobile device and able to execute the code of file creation but unable to execute the codes of matrix multiplication and N-Queens puzzle. Therefore cloudlet 1 acts as the proxy server and selects the optimum cloudlet among cloudlet 2, cloudlet 3 and cloudlet 4 which are able to execute the matrix multiplication and N-Queens puzzle codes. Cloudlet 1, cloudlet 2, cloudlet 3 and cloudlet 4 are unable to execute the code of sorting character array. Therefore this code is executed inside the cloud.

A) Offloading code of file creation to the proxy server: The proxy server is able to execute the code of file creation requested by the mobile device. The latency and the power consumption while offloading the file creation code to cloudlet 1 i.e. the proxy server are collected and presented in TABLE 5. In this case two-level offloading to the nearest cloudlet acting as proxy server occurs.

B) Offloading codes of matrix multiplication and N-Queens puzzle to the optimum cloudlet selected by proxy server: Cloudlet 1 i.e. the proxy server is unable to execute the codes of matrix multiplication and N-Queens puzzle. Cloudlet 2, cloudlet 3 and cloudlet 4 are able to execute the codes of matrix multiplication and N-Queens puzzle. The optimum cloudlet among these three is selected by Cloudlet 1 using the proposed algorithm and three-level offloading to the optimum cloudlet occurs. Two cases are considered in matrix multiplication:

Case 1: Square matrix multiplication of order 100x100.

Case 2: Non-square matrix multiplication- first matrix of order 100x200 and second matrix of order 200x300.

In N-Queens puzzle, N non-attacking queens have to be placed in an N x N board [32-33]. Non-attacking means the queens will be placed in such a way that no two queens will be in the same row, same column and same diagonal as shown in TABLE 4. This is a popular case of backtracking and is a research area in the field of computer science including neural networks, parallel memory storage schemes, deadlock prevention, parity check codes, image processing etc [32]. N-Queens puzzle does not have unique solution and its execution with different initial values give different outputs. One of the solutions of 4-Queens puzzle is shown in TABLE 4 where 'Q' denotes a queen.

TABLE 4
PLACEMENT OF QUEENS IN ONE OF THE SOLUTIONS OF 4-QUEENS PUZZLE

	Q		
			Q
Q			
		Q	

N-Queens puzzle is considered in our experiment with the following two cases:

Case 1: 4-Queens puzzle.

Case 2: 8-Queens puzzle.

The latencies and the power consumptions while offloading the matrix multiplication code and N-Queens puzzle code to cloudlet 2, cloudlet 3 and cloudlet 4 are collected and presented in TABLE 5. Based on these results the optimum cloudlet is selected. For matrix multiplication cloudlet 4 is selected as optimum with respect to low power consumption and low latency. But for N-Queens puzzle cloudlet 4 gives minimum latency whereas cloudlet 2 gives minimum power. If low power has the higher priority, then cloudlet 2 is selected. Otherwise if low latency has higher priority cloudlet 4 is selected. Hence cloudlet 2 is optimum with respect to low power consumption and cloudlet 4 is optimum with respect to low latency in this case.

C) Offloading code of character array sorting to the cloud: Cloudlet 1, cloudlet 2, cloudlet 3 and cloudlet 4 are unable to execute this code. Thus the code of sorting character array is executed inside the cloud. The latency and power consumption in offloading this code are obtained and presented in TABLE 5. In this case three-level offloading to the cloud occurs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

TABLE 5

LATENCY AND POWER CONSUMPTION IN OFFLOADING PROCESS CODE USING PROPOSED ALGORITHM

A) Two-level offloading to proxy server: Offloading code of file creation to Cloudlet 1 acting as proxy server			**Remarks**
	Latency (sec)	**Power (W)**	
Creation of ten files of size 1KB using Cloudlet 1 working as proxy server	5.009 sec Snapshot: real 0m5.009s user 0m0.000s sys 0m0.000s	0.26 W	File creation code is offloaded to proxy server
B) Three-level offloading to optimum cloudlet: Offloading codes of matrix multiplication and N-Queens puzzle to optimum cloudlet selected by proxy server			Proxy server is unable to execute matrix multiplication and N-Queens puzzle codes
i) Selection of optimum cloudlet with respect to low power plus low latency for offloading code of matrix multiplication			Proxy server searches for optimum cloudlet for offloading matrix multiplication code

Sl. no.	**Latency (sec)**		**Power (W)**		
	Order of matrix		**Order of matrix**		
	100x100_100x100	**100x200_200x300**	**100x100_100x100**	**100x200_200x300**	
Cloudlet 2	5.988 sec Snapshot: real 0m5.952s user 0m0.012s sys 0m0.024s	6.555 sec Snapshot: real 0m6.475s user 0m0.040s sys 0m0.040s	0.3 W	0.34 W	• Maximum power consumption • Maximum latency
Cloudlet 3	3.26 sec Snapshot: Time=3260 ms	4.649 sec Snapshot: Time=4649 ms	0.16 W	0.23 W	• Medium power consumption • Medium latency
Cloudlet 4	2.745 sec Snapshot: Time=2745 ms	4.181 sec Snapshot: Time=4181 ms	0.14 W	0.21 W	• **Minimum power consumption** • **Minimum latency**

ii) Selection of optimum cloudlet with respect to low power or low latency for offloading code of N-Queens puzzle					Proxy server searches for optimum cloudlet for offloading N-Queens puzzle code
Sl. no.	**Latency (sec)**		**Power (W)**		
	Number of Queen		**Number of Queen**		
	4 Queens	**8 Queens**	**4 Queens**	**8 Queens**	
Cloudlet 2	1.584 sec Snapshot: real 0m1.584s user 0m0.000s sys 0m0.000s	2.495 sec Snapshot: real 0m2.491s user 0m0.000s sys 0m0.004s	0.08 W	0.12 W	• **Minimum power consumption** • Medium latency
Cloudlet 3	2.028 sec Snapshot: Time=2028 ms	3.385 sec Snapshot: Time=3385 ms	0.11 W	0.17 W	• Maximum power consumption • Maximum latency
Cloudlet 4	1.201 sec Snapshot: Time=1201 ms	2.324 sec Snapshot: Time=2324 ms	0.1 W	0.16 W	• **Minimum latency** • Medium power consumption

C) Three-level offloading to cloud: Offloading code of sorting character array to cloud			All the available cloudlets are unable to execute character array sorting code and this code is offloaded to the cloud by the proxy server
	Latency (sec)	**Power (W)**	
Character array sorting of 25 elements	53.188 sec real 0m53.185s user 0m0.000s sys 0m0.003s	2.7 W	

Inference from experimental results:

- Cloudlet 1 executes the file creation code with the given input and sends back the result to the requested mobile device.
- Cloudlet 1 is unable to execute the matrix multiplication code and the N-Queens puzzle code and calculates the power consumptions and latencies for offloading these codes for given input to cloudlet 2, cloudlet 3 and cloudlet 4.
- Cloudlet 4 consumes approximately 36-54% and 10-16% less latency than cloudlet 2 and cloudlet 3 respectively in case of offloading the code of matrix multiplication for the given input. Cloudlet 4 consumes approximately 38-53% and 8-12% less power than cloudlet 2 and cloudlet 3 respectively in case of offloading the code of matrix multiplication for the given input. Cloudlet 1 working as proxy server selects cloudlet 4 as the optimum cloudlet with respect to low power consumption and low latency for offloading the code of matrix multiplication for the given input.
- Cloudlet 4 consumes approximately 7-24% and 31-40% less latency than cloudlet 2 and cloudlet 3 respectively in case of offloading the code of N-Queens puzzle for the given input. Cloudlet 2 consumes approximately 27-29% and 20-25% less power than cloudlet 3 and cloudlet 4 respectively in case of offloading the code of N-Queens puzzle for the given input. Cloudlet 1 working as proxy server selects cloudlet 2 as the optimum cloudlet with respect to low power consumption and cloudlet 4 as the optimum cloudlet with respect low latency for offloading the code of N-Queens puzzle for the given input.
- Cloudlet 1, cloudlet 2, cloudlet 3 and cloudlet 4 are unable to execute the code of character array sorting with the given input. Thus the cloud is used to execute this code.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

Using the experimental data presented in TABLE 5, the power consumptions and latencies in offloading matrix multiplication code to cloudlet 2, cloudlet 3 and cloudlet 4 are pictorially presented in Fig.5 and Fig.6. From Fig.5 and Fig.6 it is observed that using cloudlet 4 the power consumption and latency both are reduced than cloudlet 2 and cloudlet 3. Therefore cloudlet 4 is optimum for offloading the code of matrix multiplication. Using the experimental data of TABLE 5, the power consumptions and latencies in offloading N-Queens puzzle code to cloudlet 2, cloudlet 3 and cloudlet 4 are pictorially presented in Fig.7 and Fig.8. From Fig.7 and Fig.8 it is observed that using cloudlet 2 the power consumption is reduced than the cloudlet 3 and cloudlet 4, and using cloudlet 4 the latency is reduced than the cloudlet 2 and cloudlet 3. Therefore cloudlet 2 is optimum with respect to low power consumption and cloudlet 4 is optimum with respect to low latency for offloading the code of N-Queens puzzle.

5.2 OptimumCloudletSelect: An Android Application for Optimum Cloudlet Selection

In the MCC laboratory of our university, we have created an Android application named "OptimumCloudletSelect" for experimental purpose. This application is used to select the optimum cloudlet providing low power and low latency for offloading a job. Fig.9.(a) shows the available cloudlets. User selects the job of matrix multiplication to be offloaded as shown in Fig.9.(b). User enters the order of the matrices and the names of the files containing the matrices as observed from Fig.9.(c). As higher order matrices are considered, file names are taken as input to read the values of the matrix elements contained in the corresponding file. The nearest cloudlet is cloudlet 1 which is unable to execute the matrix multiplication code.

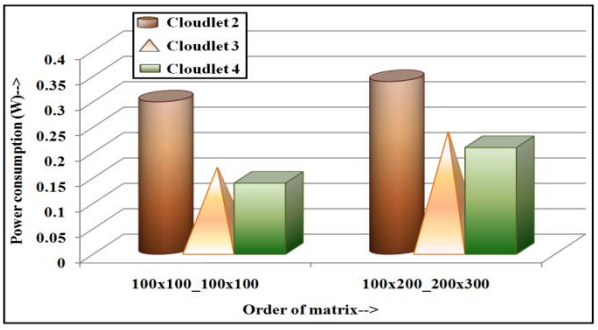


Fig.5. Power consumptions in offloading matrix multiplication code to cloudlet 2, cloudlet 3 and cloudlet 4

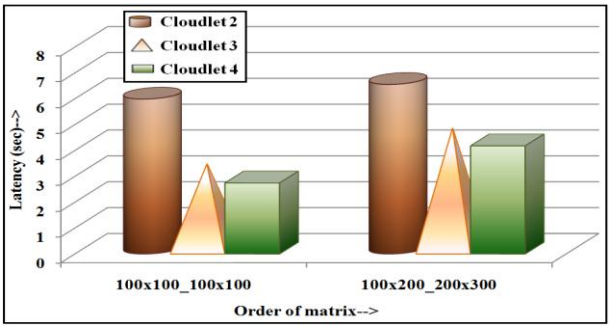


Fig.6. Latencies in offloading matrix multiplication code to cloudlet 2, cloudlet 3 and cloudlet 4

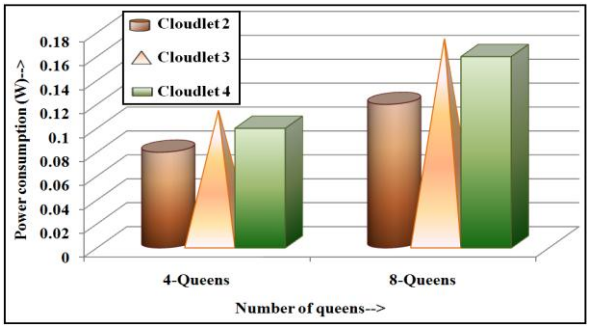


Fig.7. Power consumptions in offloading N-Queens puzzle code to cloudlet 2, cloudlet 3 and cloudlet 4

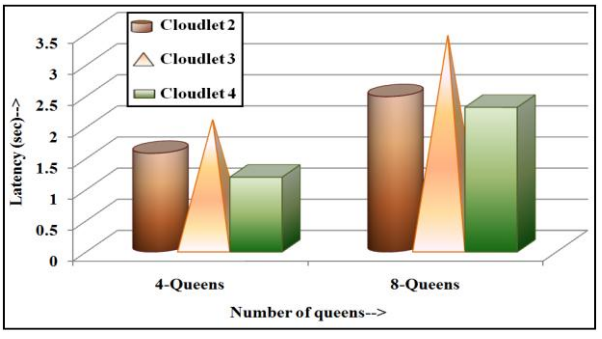


Fig.8. Latencies in offloading N-Queens puzzle code to cloudlet 2, cloudlet 3 and cloudlet 4

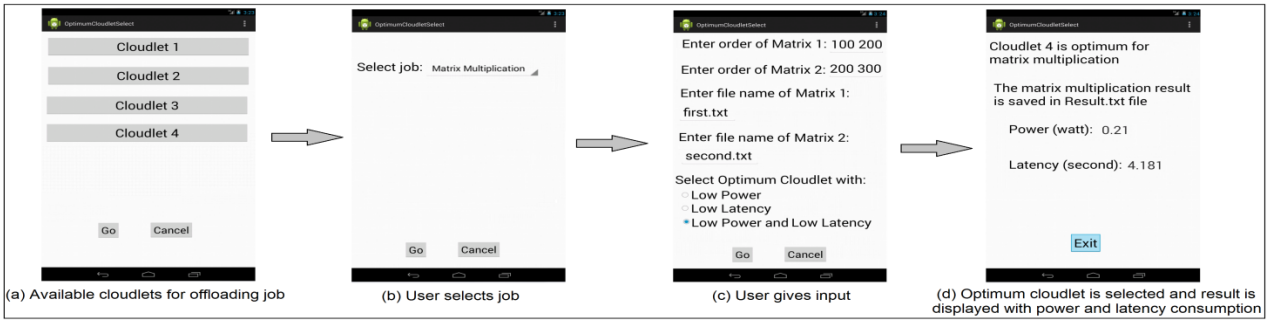


Fig.9. Offloading matrix multiplication code to optimum cloudlet with respect to low power consumption and low latency

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2586061, IEEE Transactions on Cloud Computing

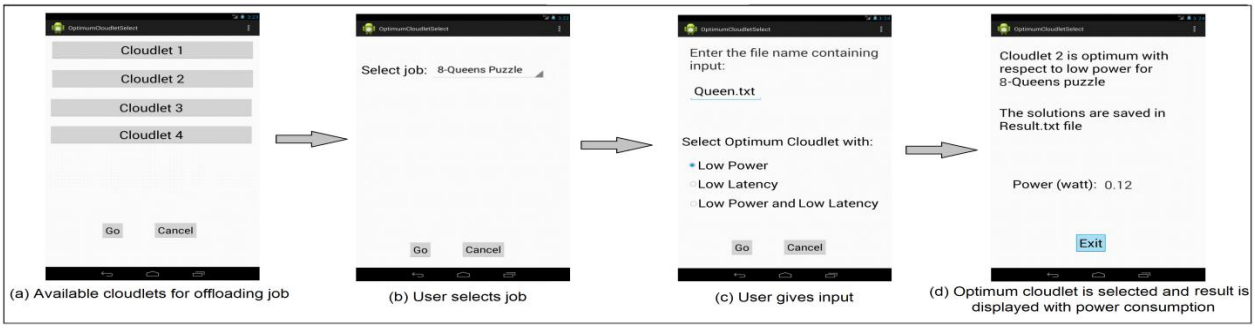


Fig.10. Offloading N-Queens puzzle code to optimum cloudlet with respect to low power consumption

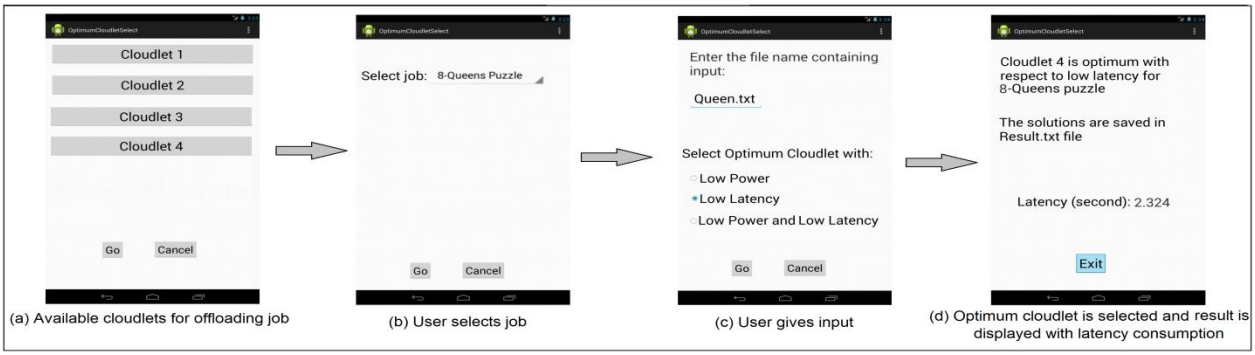


Fig.11. Offloading N-Queens puzzle code to optimum cloudlet with respect to low latency

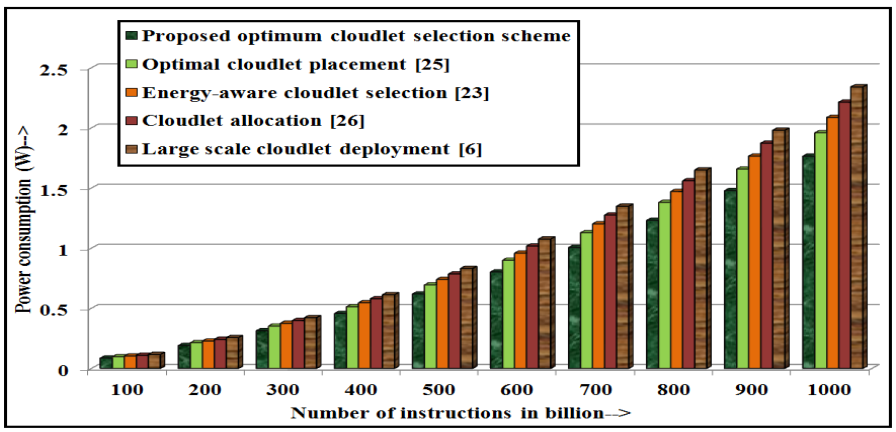


Fig.12. Comparison of power consumption between proposed and existing schemes on cloudlet allocation in multi-cloudlet environment

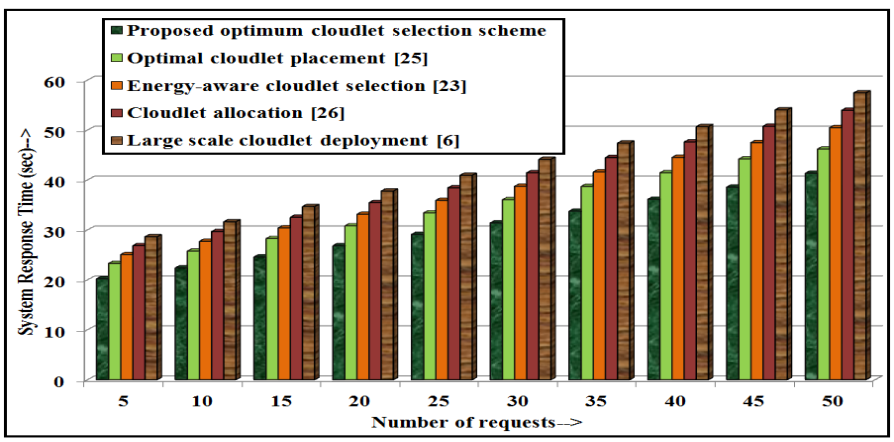


Fig.13. Comparison of system response time between proposed and existing schemes on cloudlet allocation in multi-cloudlet environment

Therefore cloudlet 1 acts as a proxy server and selects the optimum cloudlet among cloudlet 2, cloudlet 3 and cloudlet 4 with respect to low power and low latency. In this case the time of reading file is ignored and the time required in algorithm execution is evaluated to demonstrate the efficiency of the proposed approach. Cloudlet 4 consumes lower power and lower latency than cloudlet 2 and cloudlet 3. Hence cloudlet 4 is selected as optimum cloudlet as demonstrated in Fig.9.(d). The result of multiplication is saved in a file as shown in Fig.9.(d).

In Fig.10 and Fig.11 the user selects the job of 8-Queens puzzle to be offloaded. But cloudlet 1 is unable to execute this code and selects the optimum cloudlet among cloudlet 2, cloudlet 3 and cloudlet 4 with respect to low

TABLE 6
CONTRIBUTIONS AND NOVELTY OF THE PROPOSED STRATEGY WITH RESPECT TO THE EXISTING WORKS ON MULTI-CLOUDLET SCENARIO

Properties	Existing Methods on Cloudlet Allocation in Multi-cloudlet Scenario				Our Proposed Strategy
	Large scale Cloudlet Deployment [6]	Energy-aware Cloudlet Selection [23]	Optimal Cloudlet Placement [25]	Cloudlet Allocation [26]	
Working Model	A number of cloudlets are deployed and mobile devices under the coverage of a cloudlet use its service. According to user location the nearest cloudlet is selected for offloading. If no cloudlet is available nearby, the remote cloud is used.	A dynamic offloading scheme is proposed where the cloudlet allocation takes place in an energy-efficient way.	A metropolitan wide area network is considered. Cloudlets are allocated in an optimal way in the densely populated region to balance the work load and reduce the system response time.	Cloudlets are allocated to VMs considering two cases. In the first case equal number of VMs and cloudlets are considered where a single cloudlet is executed on a single VM. In the second case, the number of VMs is half the number of cloudlets.	A power and latency aware optimum cloudlet selection method is proposed for multi-cloudlet environment with the introduction of a proxy server.
Two-level offloading to nearest cloudlet occurs	✓	✓	✓	✓	✓
Three-level offloading to optimal cloudlet selected by proxy server occurs	✗	✗	✗	✗	✓
Three-level offloading to cloud occurs	✓	✓	✓	✓	✓
Reduces power and latency than cloud based offloading	✓	✓	✓	✓	✓
Reduces power and latency than nearby cloudlet based offloading where if the cloudlet fails, cloud is used for offloading	✗	✗	✗	✗	✓
Optimum cloudlet is selected based on (low-power + low-latency) among multiple cloudlets	✗	✗	✗	✗	✓
Reduction in power consumption in proposed scheme than existing schemes	25%	15%	11%	21%	Not applicable
Reduction in system response time in proposed scheme than existing schemes	28%	18%	12%	24%	Not applicable

Remarks: In all the existing approaches [6, 23, 25, 26], two-level offloading to cloudlet and three-level offloading to cloud occur. If the selected cloudlet fails to offload, the cloud is used for offloading which increases the latency. Consequently the system response time increases. But we have introduced three-level offloading to optimum cloudlet in this paper. In our approach, the nearest cloudlet is first selected. If it is unable to offload the requested code of the application, the cloudlet acts as a proxy server and for offloading the code selects the optimum cloudlet from its nearby cloudlets providing low power or low latency or both. If no response is received from the nearby cloudlets, the cloud is selected for offloading the code. Hence in our proposed scheme three-level offloading to the optimum cloudlet exists with two-level offloading to nearest cloudlet and three-level offloading to cloud. This is the novelty of the proposed approach. Introducing three-level offloading to the optimum cloudlet, the proposed scheme achieves lower power consumption as well as lower system response time than the existing schemes [6, 23, 25, 26].

power and low latency. Cloudlet 2 consumes lower power than cloudlet 3 and cloudlet 4. But cloudlet 4 consumes lower latency than cloudlet 2 and cloudlet 3. Hence cloudlet 2 is optimum with respect to low power consumption as observed from Fig.10.(d) whereas cloudlet 4 is optimum with respect to low latency as demonstrated in Fig.11.(d). The result is saved in a file as shown in Fig.10.(d) and Fig.11.(d). In this way a job can be offloaded to the optimum cloudlet using the proposed Android application OptimumCloudletSelect.

5.3 Comparison of Proposed Strategy with Existing Works

In this section the power consumption and the system response time in case of our proposed approach are compared with that of the existing approaches on cloudlet allocation in multi-cloudlet scenario. Fig.12 shows the power consumption in case of our proposed scheme determined using equation (20), and in the existing schemes with respect to the number of instructions to be executed for the process code with input. It is observed from Fig.12 that using our proposed scheme the power consumption can be reduced by approximately 11%, 15%, 21%, and 25% respectively than the schemes proposed in [25], [23], [26] and [6] respectively. Fig.13 shows the system response time in case of our proposed scheme determined using equation (16), and in the existing schemes with respect to the number of user requests. It is observed from Fig.13 that using our proposed scheme the system response time can be reduced by approximately 12%, 18%, 24% and 28% respectively than the schemes proposed in [25], [23], [26], and [6] respectively. By reducing the response time, quality of experience of the users can be improved in our scheme. In [6], the nearest cloudlet is selected to offload an application. In [23], the most energy efficient cloudlet is selected for offloading. In [25], the cloudlets are allocated to the users in a densely populated region in such a way that the load is balanced and the system response time is reduced. In [26], the cloudlets are allocated to VMs in a way that the service quality is improved in terms of execution time. But in all the existing approaches [6, 23, 25, 26], if the selected cloudlet fails to offload the application, the cloud is used for offloading. But offloading to remote cloud increases the latency. As a result the system response time gets affected. In the existing schemes two-level offloading to cloudlet and three-level offloading to cloud occur. In our approach, the nearest cloudlet is first selected. If it is unable to offload the requested code of the application, the cloudlet acts as a proxy server and selects the optimum cloudlet from its nearby cloudlets with respect to low power or low latency or both to offload the code of the application. If none of the nearby cloudlets responds, the cloud is selected for offloading the requested code. Hence in our proposed scheme three-level offloading to the optimum cloudlet is introduced along with the cases of two-level offloading to nearest cloudlet and three-level offloading to cloud. This is the novelty of the proposed method with respect to the existing schemes on multi-cloudlet scenarios [6, 23, 25, 26] which makes it better from the perspective of low power consumption and low system response time. The novelty and contributions of the proposed strategy with respect to the existing works on multi-cloudlet scenario is presented in TABLE 6. It is demonstrated from TABLE 6 that the proposed approach is a power and latency aware offloading scheme.

6 Conclusion

In this paper a strategy for selecting optimum cloudlet in a multi-cloudlet environment is proposed. A mobile device requests its nearest cloudlet to offload a code of an application with input. If the cloudlet is able to execute the requested code of the application, it sends back the result to the mobile device. Otherwise the nearest cloudlet acts as a proxy server and selects the optimum cloudlet which will consume minimum power and minimum latency among the nearby cloudlets to execute the requested code of the application for the given input. An experimental analysis of the proposed strategy is performed. The proposed strategy is compared with the existing cloudlet allocation methods for multi-cloudlet environment to show that the proposed method reduces the power consumption and the system response time than the existing schemes. Theoretical results present that using the proposed optimum cloudlet selection method the power and the latency consumption can be reduced by approximately 29-32% and 33-36% respectively than the cloud based offloading.

Acknowledgment

Authors are grateful to 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 reference no.: SR/FST/ETI-296/2011, and No.DST/INSPIRE Fellowship/2013/327 under which this paper has been completed.

References

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," Wireless Communications and Mobile Computing, vol. 13, no. 18, pp. 1587-1611, 2013.

[2] A. Mukherjee and D. De, "Low Power Offloading Strategy for Femto-Cloud Mobile Network," Engineering Science and Technology, an International Journal, vol. 19, no 1, pp. 260-270, 2016.

[3] J. Li, X. Tan, X. Chen, D. Wong, and F. Xhafa, "OPoR: Enabling Proof of Retrievability in Cloud Computing with Resource-Constrained Devices," IEEE Trans. Cloud Computing, vol. 3, no. 2, pp. 195-205, 2015.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," Pervasive Computing, IEEE, vol. 8, no. 4, pp. 14-23, 2009.

[5] Y. Jararweh, L. A. Tawalbeh, F. Ababneh, A. Khreishah, and F. Dosari, "Scalable Cloudlet-based Mobile Computing Model," Procedia Computer Science, vol. 34, pp. 434-441, 2014.
[6] L. A. Tawalbeh, Y. Jararweh, and F. Dosari, "Large Scale Cloudlets Deployment for Efficient Mobile Cloud Computing," Journal of Networks, vol. 10, no. 1, pp. 70-76, 2015.
[7] S. Banerjee, M. Adhikari, S. Kar, and U. Biswas, "Development and Analysis of A New Cloudlet Allocation Strategy for QoS Improvement in Cloud," Arabian Journal for Science and Engineering, vol. 40, no. 5, pp. 1409-1425, 2015.
[8] A. Bahtovski and M. Gusev, "Cloudlet Challenges," Procedia Engineering, vol. 69, pp. 704-711, 2014.
[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.
[10] L. Heilig and S. Voss,"A Scientometric Analysis of Cloud Computing Literature," IEEE Trans. Cloud Computing, vol. 2, no. 3, pp. 266-278, 2014.
[11] Y. H. Chiang, Y. C. Ouyang, and C. T. Hsu, "An Efficient Green Control Algorithm in Cloud Computing for Cost Optimization," IEEE Trans. Cloud Computing, vol. 3, no. 2, pp. 145-155, 2015.
[12] F. Chen, T. Xiang, X. Lei, and J. Chen, "Highly Efficient Linear Regression Outsourcing to a Cloud," IEEE Trans. Cloud Computing, vol. 2, no. 4, pp. 499-508, 2014.
[13] J. Pooyan, A. Aakash, and P. Claus, "Cloud Migration Research: A Systematic Review," IEEE Trans. Cloud Computing, vol.1, no. 2, pp. 142-157, 2013.
[14] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud," IEEE Trans. Cloud Computing, vol. 2, no. 1, pp. 14-28, 2014.
[15] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds," IEEE Trans. Cloud Computing, vol. 2, no. 2, pp. 222-235, 2014.
[16] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, H. Liu, "Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments," IEEE Trans. Cloud Computing, vol. 3, no. 2, pp. 119-131, 2015.
[17] C. W. Tsai, W. C. Huang, M. H. Chiang, M. C. Chiang, and C. S. Yang, "A Hyper-Heuristic Scheduling Algorithm for Cloud," IEEE Trans. Cloud Computing, vol. 2, no. 2, pp. 236-250, 2014.
[18] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds," IEEE Trans. Cloud Computing, vol. 2, no. 2, pp. 168-180, 2014.
[19] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile Cloud Computing: A Survey," Future Generation Computer Systems, vol. 29, no. 1, pp. 84-106, 2013.
[20] A. Mukherjee and D. De, "Femtocell Based Green Health Monitoring Strategy," URSIGA, IEEE, pp. 1-4, 2014.
[21] D. De and A. Mukherjee, "Femto-Cloud Based Secure and Economic Distributed Diagnosis and Home Health Care System," Journal of Medical Imaging and Health Informatics, vol. 5, no. 3, pp. 435-447, 2015.
[22] M. Quwaider and Y. Jararweh, "Cloudlet-based Efficient Data Collection in Wireless Body Area Networks," Simulation Modelling Practice and Theory, vol. 50, pp. 57-71, 2014.
[23] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic Energy-aware Cloudlet-based Mobile Cloud Computing Model for Green Computing," Journal of Network and Computer Applications, vol. 59, pp. 46-54, 2016.
[24] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt, "Adaptive Deployment and Configuration for Mobile Augmented Reality in The Cloudlet," Journal of Network and Computer Applications, vol. 41, pp. 206-216, 2014.
[25] M. Jia, J. Cao, and W. Liang, "Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks," IEEE Trans. Cloud Computing, 2015.
[26] M. Shiraz and A. Gani, "Mobile Cloud Computing: Critical Analysis of Application Deployment in Virtual Machines," International Proceedings of Computer Science & Information Tech, vol. 27, pp. 11-16, 2012.
[27] K. Kumar and Y. H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," Computer, vol. 4, pp. 51-56, 2010.
[28] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu, "A Stochastic Scheduling Algorithm for Precedence Constrained Tasks on Grid," Future Generation Computer Systems, vol. 27, no. 8, pp. 1083-1091, 2011.
[29] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards Energy-Efficient Scheduling for Real-Time Tasks Under Uncertain Cloud Computing Environment," Journal of Systems and Software, vol. 99, pp. 20-35, 2015.
[30] R. Surgiewicz, N. Strom, A. Ahmed, and Y. Ai., "LTE Uplink Transmission Scheme," pp. 1-7, 2014.
[31] J. Veizades and C. E. Perkins, "Service Location Protocol", 1997.
[32] J. Bell and B. Stevens, "A Survey of Known Results and Research Areas for N-Queens," Discrete Mathematics, vol. 309, no. 1, pp. 1-31, 2009.
[33] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic Resource Allocation and Parallel Execution in The Cloud for Mobile Code Offloading," Proc. IEEE INFOCOM, pp. 945-953, 2012.

Anwesha Mukherjee is currently pursuing her PhD as a DST-INSPIRE Fellow in the field of mobile network. Her research areas are green mobile network and mobile cloud computing. She has received Young Scientist Award from International Union of Radio Science in 2014 at Beijing, China. Her email id: anweshamukherjee2011@gmail.com.

Dr. Debashis De (M'13-SM'15) is an Associate Professor and presently Head of the Department of Computer Science and Engineering of West Bengal University of Technology, India and Adjunct Research Fellow of University of Western Australia, Australia. His research area includes energy and latency optimization in mobile cloud computing. He has received Young Scientist award both in 2005 at New Delhi and in 2011 at Istanbul from International Union of Radio Science, H. Q., Belgium. His email id: dr.debashis.de@ieee.org.

Deepsubhra Guha Roy is currently pursuing his PhD in the field of Mobile Cloud Computing. His research area is QoS improvement in mobile cloud computing. His email id: roysubhraguha@gmail.com.