J Supercomput (2018) 74:2412–2448
https://doi.org/10.1007/s11227-018-2269-x

C2OF2N: a low power cooperative code offloading method for femtolet-based fog network

Anwesha Mukherjee[1] · **Priti Deb**[2] · **Debashis De**[2] · **Rajkumar Buyya**[3]

Published online: 6 February 2018

Abstract Power and delay aware cloud service provisioning to mobile devices has become a promising domain today. This paper proposes and implements a cooperative offloading approach for indoor mobile cloud network. In the proposed work mobile devices register under femtolet which is a home base station with computation and data storage facilities. The resources of the mobile devices are collaborated in such a way that different mobile devices can execute different types of computations based on cooperative federation. The proposed offloading scheme is referred as cooperative code offloading in femtolet-based fog network. If none of the mobile device can execute the requested computation, then femtolet executes the computation. Use of femtolet provides the mobile devices voice call service as well as cloud service access. Femtolet is used as the fog device in our approach. The proposed model is simulated using Qualnet version 7. The simulation results demonstrate that the proposed scheme minimizes the energy by 15% and average delay up to 12% approximately than the existing scheme. Hence, the proposed model is referred as a low power offloading approach.

Keywords Fog computing · Cooperative offloading · Femtolet · Power consumption · Delay

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

[1] Department of Computer Science and Engineering, University of Engineering and Management, Plot No. III-B/5, New Town, Action Area-III, Kolkata, West Bengal 700160, India

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

[3] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Parkville, Australia

1 Introduction

In the area of advanced wireless network, energy efficiency and reduced delay are the foremost issues for the mobile users. Femtolet is a low power mobile network device for offloading computational task by mobile users at low delay [1]. Combination of femtocell [2] and cloudlet [3] are symbolized as femtolet. Femtocell is small cell home base station with coverage of 10–20 m. Under the coverage of a macrocell or a microcell, femtocells are allocated. It is used to fulfill the purpose of achieving good signal strength at indoor region with low power consumption. Smart phone incorporates some technical hitches like insufficient storage, limited battery life, inadequate processing power, etc. To meet the users' needs, mobile cloud computing (MCC) has come into the scenario [4]. MCC offers various types of cloud applications and services. But offloading to long-distance cloud server increases delay and energy consumption [5–10]. To achieve high speed offloading cloudlet is introduced in the network [5–8]. A cloudlet is a computer or collection of computers providing high-bandwidth access to cloud services [3]. Femtolet gives the femtocell like services as well as a cloud paradigm at low delay and reduced energy.

For real-time applications like healthcare services delay is a vital parameter. Most of tasks in such an emergency application have hard deadline. Delegating such tasks to remote cloud server increases the delay which in turn fails to meet the deadline. Hence, the quality of service (QoS) degrades. If the deadline is missed, the user is not satisfied with the service. Accordingly, the quality of user experience is affected. To overcome such difficulties, fog computing comes where offloading takes place inside the intermediate devices between the requesting device and remote cloud server. The intermediate device can be mobile device, cloudlet, etc., which is able to execute the task. Fog computing is a decentralized computing architecture where application services and computing resources are scattered at any point beside the range from the data source to the cloud [11–13]. The main aim of fog computing is to maximize the efficiency in terms of time and energy, and to minimize the amount of data that requires to be offloaded to the cloud for analysis, processing and storage. Cloud service access needs high bandwidth for transporting the data. Fog computing solves the issue by keeping the data closer than that of a cloud server. Fog computing is an alternative way to cloud computing. It helps to reduce the energy consumption at the time of data offload. Fog computing makes the strategy more energy efficient.

1.1 Motivations and contributions

In this work, we have used femtolet and fog computing to reduce the power consumption and delay in offloading. The motivations for our work are:

1. There may be a case where a number of mobile devices register under a home base station and are closely located to each other. Each of the devices has limited ability of computation. Our aim is to utilize their computation ability through a paradigm where these devices will contribute their resources such that each of them is able to execute a specific type of computation like numerical function, game, image processing, etc. In that case these devices will work as edge devices and offload

between each other through cooperative federation for utilizing their resources and reducing communication delay.
2. The mobile devices communicate with the cloudlet or the remote cloud server via an access point or a cellular base station. Hence, the mobile device has to offload its data or computation via the cellular base station or the access point. If the base station itself has the potential of executing computation intensive task and storing large volume of data, then the communication delay can be reduced. In our previous work for this purpose we have proposed femtolet which is an indoor base station providing data and computation offloading facility. Our aim is to use femtolet as a fog device. The mobile devices will offload their computations to the femtolet if the edge devices are unable to execute the requested computation for reducing communication delay.

In the proposed work under the coverage of a femtolet, code offloading takes place in a fog computing environment. Figure 1 shows the fog computing environment, where every device is able to take part in computational offloading. The mobile devices combine their resources to offload different types of computations. Each device can execute a particular type of application. The IDs of mobile devices along with the respective executable application types are stored in a table. This table is maintained by each mobile device. If a mobile device has to offload any computation, a nearby mobile device which is able to execute that type of application is selected and the required computation is executed inside that mobile device. Here offloading happens between two mobile devices registered under the same femtolet. The mobile device which performs the computation is referred as the edge device. If none of the edge device can offload it, then the femtolet serves as a fog device and executes the computation. Use of femtolet provides the mobile devices voice call service as well as internet access. The fog device femtolet is connected with the remote cloud server.

The key contributions of this paper are:

1. A cooperative offloading method is proposed based on femtolet and fog computing, where offloading occurs from one mobile device to an edge or fog device in an indoor region. The proposed strategy is referred as cooperative code offloading in femtolet-based fog network (C2OF2N).
2. In C2OF2N mobile devices register under a femtolet and collaborate by contributing their resources in such a way that each of them is able to execute specific type of application. These mobile devices work as edge devices and offload between each other through cooperative federation. The femtolet serves as fog device where the mobile devices offload their computations if edge devices are unable to execute the requested code.
3. The power consumption and delay in C2OF2N are determined and compared with existing offloading schemes.
4. The simulation results show that using C2OF2N delay and energy consumption are reduced than the existing scheme.

The rest of the paper is structured as follows. The related work is discussed in Sect. 2, and our proposed approach C2OF2N is presented in Sect. 3. The power consumption and delay in C2OF2N are discussed in Sect. 4, analyses of the performance of C2OF2N in Sect. 5, and finally we conclude our paper in Sect. 6.

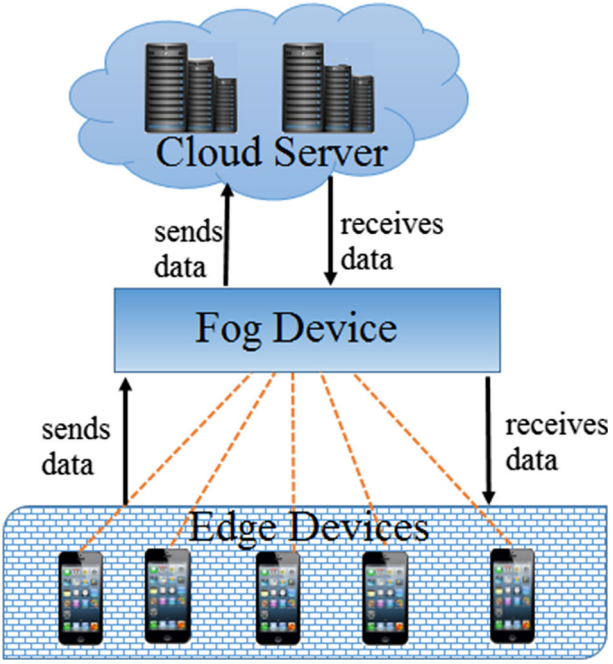

Fig. 1 Fog network formation using edge devices, fog device and remote cloud server

2 Related work

Computationally, mobile devices are confined with respect to large scale servers based on their battery life and storage capacity [14]. Cloud computing becomes essential in the field of energy-efficient mobile communication [15] because computational part is done in the remote cloud servers by offloading the applications from the mobile devices to the cloud servers [16]. A model for resource scheduling has been proposed for cloud environment to improve the QoS in [17]. To handle resource intensive mobile applications in an energy-efficient manner mobile cloud computing becomes popular [18]. Offloading of elastic mobile applications is a vital phenomenon. For runtime profiling and partitioning of such applications an approach has been proposed in [19]. Most of the researches are carried out on virtualized application offloading. A native code offloading-based multimedia framework is illustrated in [20]. Context-aware computation offloading is discussed in [21]. It is an on-demand context-aware offloading strategy which dynamically selects the resources to offload. For minimizing delay of long-distance cloud access, cloudlet has come into the scenario [3]. Cloudlet provides data offloading and task delegation facility to the mobile devices connected with it [5]. For offloading at low energy, a dynamic offloading scheme for multi-cloudlet environment is discussed in [6]. To find the optimum cloudlet in multi-cloudlet environment, a method is proposed in our previous work [7]. In this method a proxy server takes the decision of offloading to a particular cloudlet at minimum power consumption and latency. Application aware offloading is discussed in [8]. In that work various

cloudlets are enabled to offload different types of applications. Sometimes nearby mobile devices are also able to execute some task. In that case use of cloudlet or cloud server causes delay. For such a case, energy-efficient cooperative offloading model (E2COM) is discussed in [9]. Here, the mobile devices communicate with the remote cloud server through different access points; a mobile device under an access point delegates its task to another mobile device belonging under the coverage of another access point to avoid the delay of communication with remote cloud server. But if the requested device fails, then cloud server is used.

Now when a mobile device communicates it has to register under a base station. For indoor region mobile devices are registered under the home base station femtocell. When a mobile device offloads to a cloudlet, it has to communicate via the femtocell. Thus, communication delay is involved between femtocell and cloudlet. To overcome this problem femtolet has been proposed in [1]. Femtolet provides low latency and low power cloud service access with communication facility.

For providing high speed and energy-efficient cloud service to the mobile devices fog computing is introduced [11–13]. In a fog computing environment instead of accessing a remote cloud server nearby devices are used for data and computation offloading. Fog computing is mainly popular in the domain of Internet of things (IoT). In fog computing sensors and actuators are used to sense the object status, and the computation and data processing take place inside the devices between the end points and cloud server. When the computation takes place inside a mobile edge device, it is called mobile edge computing [22,23]. It is a promising area of research to fulfill the increasing computational requirements from smart phones. Rigorous applications are offloaded to the edge devices to increase the QoS. In [24] an energy-efficient mobile edge computing system with energy harvesting devices is investigated for offloading purposes. Another multiuser computational scheme is discussed in [25] for mobile edge computing. Here, multi-channel cellular interference paradigm has considered. A game theoretical approach has chosen in [25] for offloading applications based on mobile edge computing.

In our present work we have proposed a femtolet-based fog network where mobile devices register under a femtolet and work as edge devices to offload code between themselves through cooperative offloading. The femtolet works as fog device in the present work.

3 Cooperative code offloading in femtolet-based fog network

The cooperative code offloading in femtolet-based fog network, i.e., C2OF2N, is proposed in this section. The proposed method is based on femtolet and fog computing as follows:

- Femtolet is a home base station which possesses storage and has the ability of computation. Femtolet provides data and code offloading service to the mobile devices registered under it. In C2OF2N femtolet works as the fog device.
- In C2OF2N mobile devices register under a particular femtolet and collaborates their resources in such a way that each mobile device can execute a particular type of application. Here, a mobile device can offload a code related to an application to

another mobile device based on the type of application to be offloaded. The mobile device executing the code is referred as edge device. Hence in the proposed strategy, the mobile devices form a cooperative network under the femtolet. These mobile devices are edge devices of the network. These edge devices offload their codes between themselves based on cooperative federation.

- In C2OF2N different mobile devices are able to execute different types of applications. As the mobile devices form a cooperative network, one mobile device offloads code to another mobile device based on cooperative federation. For example, one mobile device is able to execute codes of game-related application and another mobile device is able to execute codes of numerical operation. When the first one has to execute code of numerical operations, it will request the second one. Similarly when the second one has to execute code of a game, it will request the first one. Thus, cooperation takes place.

The considerations in the proposed method are as follows:

- Mobile devices register under a femtolet which is the fog device. Femtolet is connected with the remote cloud server.
- Mobile devices registered under the same femtolet form a group and collaborates their resources. Here, each device has the ability to execute a particular type of application.
- Let there are N mobile devices $\{M_1, M_2, \ldots, M_N\}$ registered under a femtolet. These N devices form a group by collaborating their resources.
- Let the set of executable applications by the mobile devices is $A = \{A_1, A_2, \ldots, A_N\}$ where A_i is executable by the mobile device M_i.
- Let each application (A_i) has a set of codes denoted as $A_i = \{C_1, C_2, \ldots, C_k\}$, where k is the number of codes for application A_i.
- When a mobile device M_i needs to offload a code of an application, it selects another mobile device from rest of the $N-1$ mobile devices based on the application type. The selected mobile device is the edge device with respect to the requesting mobile device. Then, cooperation is formed between the requesting mobile device and the edge device.

The ID of each mobile device belonging to the group along with the executable type of application is stored inside a table referred as Device Information Table (DIT) as shown in Table 1. DIT is maintained by each mobile device in the group and updated at a fixed time interval T. The time interval for updating the DIT is neither very small nor large. The time interval should be such that all mobile devices can update their tables, and the method is scalable. Table 1 shows that different mobile devices are able to execute different categories of applications based on the configuration of the device, i.e., operating system (OS) and architecture, which considers the details of RAM, HDD size and processor.

Table 1 Format of DIT

Mobile device	Configuration		Executable application type
	OS	Architecture	
M_1	Windows/Linux	RAM, HDD, Processor details	A_1
M_2	Windows/Linux	RAM, HDD, Processor details	A_2
…	…	…	…
…	…	…	…
M_n	Windows/Linux	RAM, HDD, Processor details	A_n

The proposed code offloading algorithm is presented as follows:

Algorithm 1: Code offloading in femtolet based fog network

Start
When a mobile device M_m has to offload a code C_j of application A_p, the mobile device checks whether $m=p$
If $m=p$,
    M_m executes C_j locally //Local execution
Else
    M_m accesses the DIT and checks whether $A_p \in A$
    If $A_p \in A$,
        M_m selects the mobile device M_p as the edge device for offloading
        M_m sends a request to M_p for execution of C_j
        M_p executes the code and returns the result to M_m //Offloading to edge device
    Else
        M_m forwards the request to the fog device femtolet //Offloading to fog device
        If the femtolet is unable to execute the code,
            The femtolet forwards the code to the cloud server
            The cloud server executes the code and sends back the result to the femtolet
            The femtolet returns the result to M_m
        Else
            The femtolet executes the code and sends back the result to M_m
        End if
    End if
End if
End

C2OF2N is pictorially presented in Fig. 2a where four mobile devices register under a femtolet F. Mobile devices register under the femtolet using Home Node B Application Part (HNBAP) protocol. The connection protocol used between mobile devices is IPV6. The mobile devices connect with the femtolet through Uu interface. This interface connects Universal Terrestrial Radio Access Network with user equipment. The femtolet is connected with Home Node Base Station-Gateway (HNB-GW) via Internet Security Gateway using Iuh interface. The HNB-GW connects to the core network via Iucs/Iups interface. In Fig. 2a the interconnection between the devices registered under the fog device femtolet F is shown along with the interfaces. These devices M_1, M_2, M_3 and M_4 form a group. These devices are interconnected, and each of the devices can offload code to another device in the group based on the application type of the code to be executed. In Fig. 2b the code offloading process is shown.

In Fig. 2b three following cases are depicted:

Case 1 *Code offloading to edge device* A mobile device M_1 has to offload a code of an application. M_1 checks DIT and finds that this application is executable by the edge

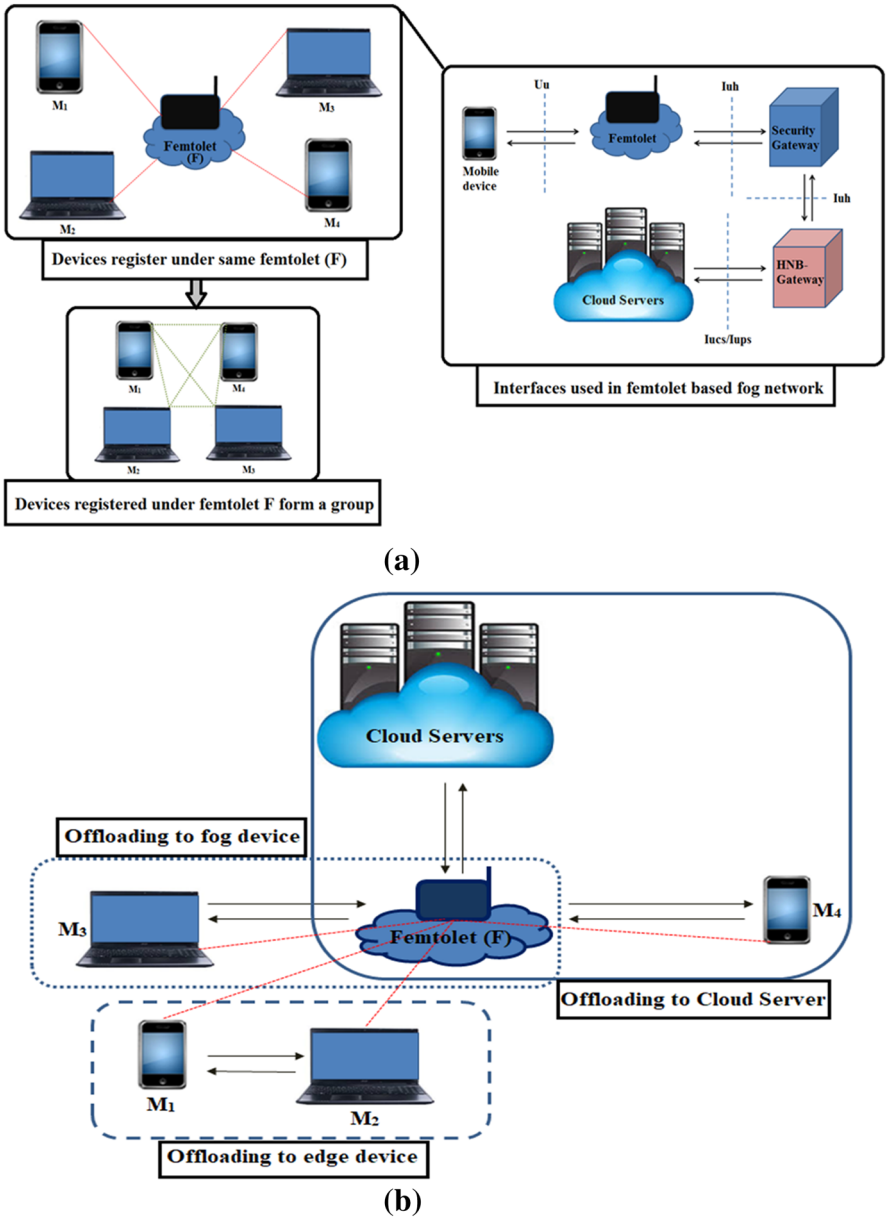

Fig. 2 Proposed femtolet-based fog network for cooperative code offloading. **a** Interconnection between mobile devices registered under a femtolet, **b** code offloading to edge device, fog device and cloud servers

device M_2. Hence, M_1 requests M_2 for executing that code. M_2 executes the code and returns the result to M_1. In this case offloading occurs from a mobile device to an edge device.

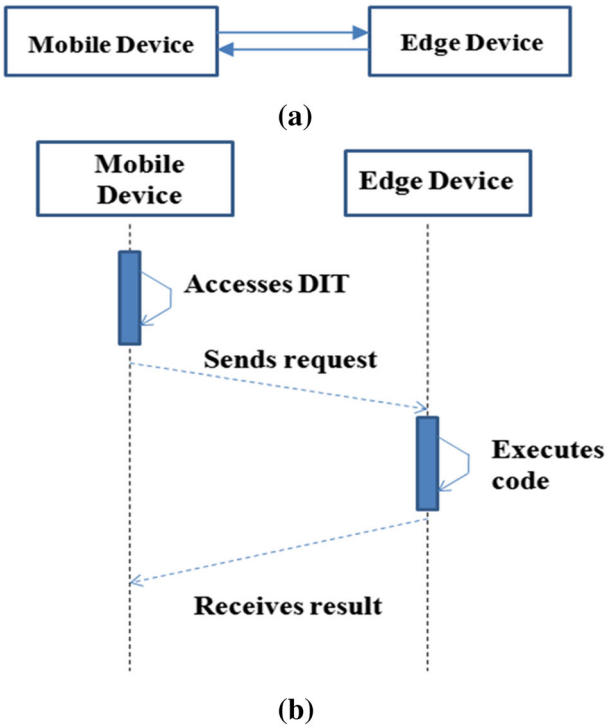

Fig. 3 Class and sequence diagram for case 1. **a** Class diagram, **b** sequence diagram

Case 2 *Code offloading to fog device* In this case, a mobile device (M_3) has to offload a code related to an application. M_3 checks DIT and finds that this application does not belong to the set of applications executable by other mobile devices. Hence, M_3 sends the request to the fog device femtolet (F). The femtolet after execution returns the result to M_3.

Case 3 *Case 3: Code offloading to cloud server via fog device* In this case, a mobile device (M_4) has to offload a code related to an application. M_4 checks DIT and finds that this application does not belong to the set of applications executable by other mobile devices. Hence, M_4 sends the request to the fog device femtolet. The femtolet is also unable to execute that code. Thus, the femtolet forwards the request to the cloud server. The cloud server after execution sends the result to the fog device femtolet, which forwards the result to M_4.

3.1 Sequence diagram

3.1.1 Sequence diagram for code offloading to edge device

Figure 3 shows sequence diagram and class diagram for case 1, where a mobile device offloads a code to another mobile device which serves as an edge device.

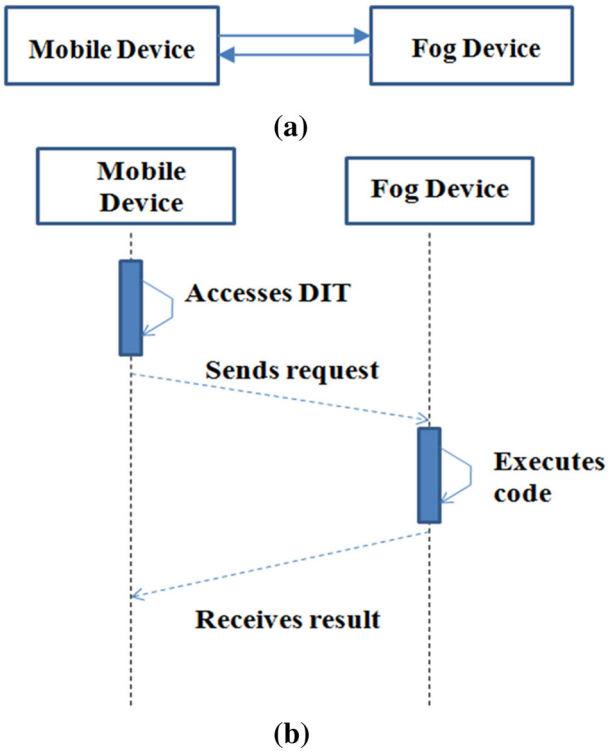

Fig. 4 Class and sequence diagram for case 2. **a** Class diagram, **b** sequence diagram

In case 1, the classes are mobile device and edge device. In case 1, the mobile device accesses the DIT and based on the application type selects the edge device. The mobile device sends request to that edge device for offloading the code. The edge device executes the code and returns the result to the mobile device.

3.1.2 Sequence diagram for code offloading to fog device

Figure 4 shows sequence diagram and class diagram for case 2, where a mobile device offloads a code to the fog device femtolet.

In case 2, the mobile device accesses the DIT. None of the edge device can execute the requested type of application. So the mobile device sends request to the fog device. The fog device executes the code and returns the result to the mobile device.

3.1.3 Sequence diagram for code offloading to cloud server via fog device

Figure 5 shows sequence diagram and class diagram for case 3, where a mobile device offloads a code to the cloud server via the fog device. In case 3, the mobile device accesses the DIT and finds that none of the edge device can execute the requested type of application. So the mobile device sends request to the fog device femtolet. The fog

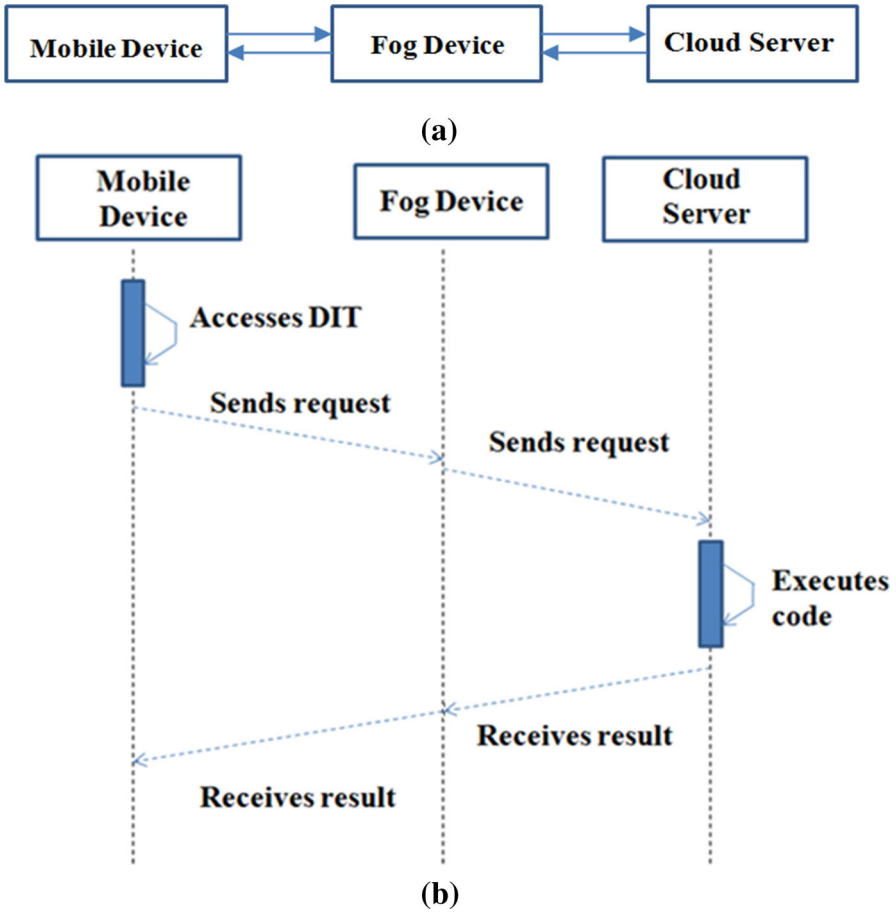

Fig. 5 Class and sequence diagram for case 3. **a** Class diagram, **b** sequence diagram

device also cannot execute that application. Hence, the fog device forwards request to the cloud server, which executes the code and returns the result to the fog device. The fog device then forwards the result to the requesting mobile device.

3.2 Application program offloading in femtolet-based fog network

In Sect. 3.1 we have discussed on the scenarios where a mobile device requests an edge device or a fog device to offload a code of an application. In this section we will discuss the full and partial offloading of application program in femtolet-based fog network. An application program is composed of a set of codes. It may occur that a mobile device or an edge device cannot be able to execute all the codes of an application program due to its resource limitation or low battery during that time. It may also possible that the fog device is unable to execute all the codes of an application program, but able to execute some of the codes. In these cases partial application offloading will occur,

where the application program will be segmented into two parts; each part will contain a subset of codes.

If one segment is executed by the mobile device and other segment by the fog device, then partial offloading to fog device occurs. If one segment is executed by the edge device and other segment by the fog device, then partial offloading to both the edge and fog devices occurs. If one segment is executed by the fog device and other segment by the cloud server, then partial offloading to both the fog device and cloud server occurs. The application program offloading algorithm is proposed as follows:

Algorithm 2: Application Program Offloading in Femtolet Based Fog Network

```
Start
When a mobile device M_m has to offload an application program A_p, the mobile device checks whether m=p
If m=p,
    If M_m can execute A_p partially,
        A_p is segmented into two parts A_pm and A_pr, where A_pm is to be executed locally and A_pr is to be offloaded
        M_m executes A_pm locally and asks the fog device femtolet to execute A_pr //Partial local execution
        The femtolet executes A_pr and returns the result to M_m  //Partial offloading to fog device
        M_m integrates the result of local execution with the received result to produce the final result
    Else
        M_m executes A_p locally and produces result    //Complete local execution
    End if
Else
    M_m accesses the DIT and checks whether A_p ∈ A
    If A_p ∈ A,
        M_m selects the mobile device M_p as the edge device for offloading
        M_m sends a request to M_p for execution of A_p
        If M_p can execute A_p partially,
            A_p is segmented into two parts A_pe and A_pr, where A_pe is to be executed by M_p and A_pr is to be offloaded
            M_p executes A_pe and asks femtolet to execute A_pr      //Partial offloading to edge device
            The femtolet executes A_pr and returns the result to M_p    //Partial offloading to fog device
            M_p integrates the result of execution of A_pe with the received result to produce the final result
        Else
            M_p executes the application fully and produces result    //Full offloading to edge device
        End if
        The edge device returns the result to M_m
    Else
        M_m forwards the request to the femtolet which is the fog device
        If the femtolet can execute A_p partially,
            A_p is segmented into two parts A_pf and A_pr, where A_pf is to be executed by femtolet and A_pr is to be
            offloaded to cloud server
            The femtolet executes A_pf and asks the cloud server to execute A_pr   //Partial offloading to fog device
            The cloud server executes A_pr and sends the result to the femtolet //Partial offloading to cloud server
            The femtolet integrates the result of execution of A_pf with the received result to produce the final
            result
        Else if the femtolet can execute A_p fully,
            The femtolet executes A_p    //Full offloading to fog device
        Else
            The femtolet asks the cloud server to execute A_p    //Full offloading to cloud server
            After execution the cloud server sends the result to the femtolet
        End if
        The femtolet forwards the result to M_m
    End if
End if
End
```

From algorithm 2 it is observed that the application program fully executed by the edge device or fog device or the remote cloud server. If a set of codes is executed by the mobile device locally and rest of the codes are executed by the fog device, then it is a case of partial offloading to fog device. If a set of codes is executed by the edge device and rest of the codes are executed by the fog device, then this is a case of partial

offloading to edge device as well as fog device. If a set of codes is executed by the fog device and rest of the codes are executed by the cloud server, then this is a case of partial offloading to fog device as well as cloud server. The different types of partial offloading scenarios of application program are illustrated in algorithm 2.

4 Power consumption and delay

4.1 Parameters used

The parameters used in calculating delay and power are presented in Table 2.

4.2 Delay

At first the delay in offloading a single code is calculated, and then the delay in application program offloading is determined.

4.2.1 Delay in offloading a code to edge device

If a mobile device offloads a code C_k to an edge device, the total delay for communication in uplink and downlink is given by,

$$L_{case1\_comk} = (Da_{uk1}/D_u) + (Da_{dk1}/D_d) \tag{1}$$

In case 1, the propagation delay for offloading a code C_k is given by,

$$L_{case1\_prok} = D_{me}/S_{pro} \tag{2}$$

In case 1, the delay for executing a code C_k inside an edge device is given by,

$$L_{case1\_cmpk} = I_k/S_e \tag{3}$$

Hence, the total delay for offloading a code C_k from a mobile device to an edge device is given as,

$$L_{case1k} = L_{case1\_comk} + L_{case1\_prok} + L_{case1\_cmpk} + T_{a1} \tag{4}$$

4.2.2 Delay in offloading a code to fog device

If a mobile device offloads a code C_k to the fog device, the total delay for communication in uplink and downlink is given by,

$$L_{case2\_comk} = (Da_{uk2}/D_u) + (Da_{dk2}/D_d) \tag{5}$$

Table 2 Parameters used for delay and power calculation in C2OF2N

Parameter	Definition
P_{mi}	Power consumption by a mobile device per unit time when offloading occurs
P_{ts}	Power consumption by a mobile device per unit time while sending data
P_{tr}	Power consumption by a mobile device per unit time while receiving data
S_m	Speed of the mobile device
S_e	Speed of the executing edge device
S_{pro}	Propagation speed
S_f	Speed of fog device femtolet
S_{cl}	Speed of cloud server
D_{me}	Distance between the requesting mobile device and the executing edge device
D_{mf}	Distance between the requesting mobile device and the fog device
D_{fcl}	Distance between the fog device and the cloud server
D_u	Uplink data transmission rate
D_d	Downlink data transmission rate
$Da_{uk}1$	Data amount transmitted in uplink for offloading a code C_k to edge device
$Da_{dk}1$	Data amount transmitted in downlink for offloading a code C_k to edge device
$Da_{uk}2$	Data amount transmitted in uplink for offloading a code C_k to fog device
$Da_{dk}2$	Data amount transmitted in downlink for offloading a code C_k to fog device
$Da_{uk}3$	Data amount transmitted in uplink for offloading a code C_k to cloud server via fog device
$Da_{dk}3$	Data amount transmitted in downlink for offloading a code C_k to cloud server via fog device
I_k	Number of instructions to be executed for the code C_k
T_a1	DIT access time if offloading occurs to edge device
T_a2	DIT access time if none of the edge device can execute the code
P_a	Power consumption of the mobile device per unit time during DIT access

In case 2, the propagation delay for offloading a code C_k to the fog device is given by,

$$L_{case2\_prok} = D_{mf}/S_{pro} \tag{6}$$

In case 2, the delay for executing a code C_k inside the fog device is given by,

$$L_{case2\_cmpk} = I_k / S_f \tag{7}$$

Hence, the total delay for offloading a code C_k from a mobile device to the fog device is given as,

$$L_{case2k} = L_{case2\_comk} + L_{case2\_prok} + L_{case2\_cmpk} + T_{a2} \tag{8}$$

4.2.3 Delay in offloading a code to cloud server via fog device

If a mobile device offloads a code C_k to the cloud server via the fog device, the total delay for communication in uplink and downlink is given by,

$$L_{case3\_comk} = (Da_{uk3}/D_u) + (Da_{dk3}/D_d) \tag{9}$$

In case 3, the delay for propagation for offloading a code C_k to the cloud server via the fog device is given by,

$$L_{case3\_prok} = (D_{mf}/S_{pro}) + (D_{fcl}/S_{pro}) \tag{10}$$

In case 3, the delay for executing a code C_k inside the cloud server is given by,

$$L_{case3\_cmpk} = I_k / S_{cl} \tag{11}$$

Hence, the total delay for offloading a code C_k from a mobile device to the cloud server via the fog device is given as,

$$L_{case3k} = L_{case3\_comk} + L_{case3\_prok} + L_{case3\_cmpk} + T_{a2} \tag{12}$$

4.2.4 Delay in offloading a full application program

In case full offloading of an application program, all the codes belonging to that application are executed inside either the edge device or fog device or remote cloud server.

In case of offloading all the codes to an edge device, the delay is given by,

$$L_{edge} = \sum_{C_k} L_{case1\_comk} + \sum_{C_k} L_{case1\_prok} + \sum_{C_k} L_{case1\_cmpk} + T_{a1} \tag{13}$$

In case of offloading all the codes to the fog device, the delay is given by,

$$L_{fog} = \sum_{C_k} L_{case2\_comk} + \sum_{C_k} L_{case2\_prok} + \sum_{C_k} L_{case2\_cmpk} + T_{a2} \tag{14}$$

In case of offloading all the codes to the cloud server via the fog device, the delay is given by,

$$L_{cloud} = \sum_{C_k} L_{case3\_comk} + \sum_{C_k} L_{case3\_prok} + \sum_{C_k} L_{case3\_cmpk} + T_{a2} \tag{15}$$

Let the number of requests arrived for offloading a full application to the edge device, fog device and cloud server are r_1, r_2 and r_3, respectively. Then, the delay in the proposed strategy considering all the requests is given as,

$$L_{offload\_app} = \frac{r_1}{r} \cdot L_{edge} + \frac{r_2}{r} \cdot L_{fog} + \frac{r_3}{r} \cdot L_{cloud} \tag{16}$$

where r is the total number of requests given by $(r_1 + r_2 + r_3)$.

4.2.5 Delay in offloading partially an application program

In partial offloading of an application program the following cases are possible:

Scenario 1 Some codes are executed inside the mobile device locally, and rest of the codes of that application program are executed inside the fog device.

Scenario 2 Some codes are executed inside the edge device, and rest of the codes of that application program are executed inside the fog device.

Scenario 3 Some codes are executed inside the fog device, and rest of the codes of that application program are executed inside the cloud server.

Delay in scenario 1

If a set of code C_{mob} execute locally inside the mobile device and another set of code C_{fog} execute inside the fog device, the delay is given by,

$$\begin{aligned} L_{partScenario1} = {} & \textstyle\sum_{C_k \in C_{mob}} (I_k / S_m) + \sum_{C_k \in C_{fog}} L_{case2\_comk} \\ & + \textstyle\sum_{C_k \in C_{fog}} L_{case2\_prok} + \sum_{C_k \in C_{fog}} L_{case2\_cmpk} + T_a \end{aligned} \tag{17}$$

where T_a is the DIT access time if the mobile device itself can execute some of the codes of the application program.

Delay in scenario 2

If a set of code C_{edge} execute inside the edge device and another set of code C_{fog} execute inside the fog device, the delay is given by,

$$\begin{aligned} L_{partScenario2} = {} & \textstyle\sum_{C_k \in C_{edge}} L_{case1\_comk} \\ & + \textstyle\sum_{C_k \in C_{edge}} L_{case1\_prok} + \sum_{C_k \in C_{edge}} L_{case1\_cmpk} \\ & + \textstyle\sum_{C_k \in C_{fog}} [(Da_{ukef} / D_u) + (Da_{dkef} / D_d)] \\ & + \textstyle\sum_{C_k \in C_{fog}} (D_{ef} / S_{pro}) + \sum_{C_k \in C_{fog}} (I_k / S_f) + T_{a1} \end{aligned} \tag{18}$$

where Da_{ukef}, Da_{dkef} are the amount of data transmission between edge and fog device in uplink and downlink, respectively, and D_{ef} is the distance between the edge and fog device.

Delay in scenario 3

If a set of code C_{fog} execute inside the fog device and another set of code C_{cloud} execute inside the cloud server, the delay is given by,

$$\begin{aligned} L_{partScenario3} = & \sum_{C_k \in C_{fog}} L_{case2\_comk} + \sum_{C_k \in C_{cloud}} L_{case3\_comk} \\ & + \sum_{C_k \in C_{fog}} L_{case2\_prok} \\ & + \sum_{C_k \in C_{cloud}} L_{case3\_prok} \\ & + \sum_{C_k \in C_{fog}} L_{case2\_cmpk} + \sum_{C_k \in C_{cloud}} L_{case3\_cmpk} + T_{a2} \end{aligned} \tag{19}$$

If the number of requests arrived for offloading in scenario 1, 2 and 3 is r_{s1}, r_{s2} and r_{s3}, respectively, the delay is given as,

$$\begin{aligned} L_{offload\_partapp} = & \frac{r_{s1}}{r_s} \cdot L_{partScenario1} + \frac{r_{s2}}{r_s} \cdot L_{partScenario2} \\ & + \frac{r_{s3}}{r_s} \cdot L_{partScenario3} \end{aligned} \tag{20}$$

where r_s is the total number of requests given by $(r_{s1} + r_{s2} + r_{s3})$.

4.3 Power consumption

At first the power consumption in offloading a single code is calculated, and then the power consumption in application program offloading is determined.

4.3.1 Power consumption in offloading a code to edge device

If a mobile device offloads a code C_k to an edge device, the total power consumed during communication in uplink and downlink is given by,

$$P_{case1\_comk} = P_{ts} \cdot (Da_{uk1}/D_u) + P_{tr} \cdot (Da_{dk1}/D_d) \tag{21}$$

In case 1, the power consumed during propagation while offloading a code C_k is given by,

$$P_{case1\_prok} = P_{mi} \cdot (D_{me}/S_{pro}) \tag{22}$$

In case 1, the power consumed while executing a code C_k inside an edge device is given by,

$$P_{case1\_cmpk} = P_{mi} \cdot (I_k/S_e) \tag{23}$$

Hence, the total power consumed for offloading a code C_k from a mobile device to an edge device is given as,

$$P_{case1k} = P_{case1\_comk} + P_{case1\_prok} + P_{case1\_cmpk} + P_a \cdot T_{a1} \tag{24}$$

4.3.2 Power consumption in offloading a code to fog device

If a mobile device offloads a code C_k to the fog device, the total power consumed during communication in uplink and downlink is given by,

$$P_{case2\_comk} = P_{ts} \cdot (Da_{uk2}/D_u) + P_{tr} \cdot (Da_{dk2}/D_d) \tag{25}$$

In case 2, the power consumed during propagation while offloading a code C_k to the fog device is given by,

$$P_{case2\_prok} = P_{mi} \cdot (D_{mf}/S_{pro}) \tag{26}$$

In case 2, the power consumed while executing a code C_k inside the fog device is given by,

$$P_{case2\_cmpk} = P_{mi} \cdot (I_k/S_f) \tag{27}$$

Hence, the total power consumed for offloading a code C_k from a mobile device to the fog device is given as,

$$P_{case2k} = P_{case2\_comk} + P_{case2\_prok} + P_{case2\_cmpk} + P_a \cdot T_{a2} \tag{28}$$

4.3.3 Power consumption in offloading a code to cloud server via fog device

If a mobile device offloads a code C_k to the cloud server via the fog device, the total power consumed during communication in uplink and downlink is given by,

$$P_{case3\_comk} = P_{ts} \cdot (Da_{uk3}/D_u) + P_{tr} \cdot (Da_{dk3}/D_d) \tag{29}$$

In case 3, the power consumed during propagation for offloading a code C_k to the cloud server via the fog device is given by,

$$P_{case3\_prok} = P_{mi} \cdot [(D_{mf}/S_{pro}) + (D_{fcl}/S_{pro})] \tag{30}$$

In case 3, the power consumed while executing a code C_k inside the cloud server is given by,

$$P_{case3\_cmpk} = P_{mi} \cdot (I_k/S_{cl}) \tag{31}$$

Hence, the total power consumed for offloading a code C_k from a mobile device to the cloud server via the fog device is given as,

$$P_{case3k} = P_{case3\_comk} + P_{case3\_prok} + P_{case3\_cmpk} + P_a \cdot T_{a2} \tag{32}$$

4.3.4 Power consumption in offloading a full application program

In case full offloading of an application program, all the codes belonging to that application are executed either inside the edge device or fog device or remote cloud server.

In case of offloading all the codes to an edge device, the power consumption is given by,

$$P_{edge} = \sum_{C_k} P_{case1\_comk} + \sum_{C_k} P_{case1\_prok} + \sum_{C_k} P_{case1\_cmpk} + P_a \cdot T_{a1} \tag{33}$$

In case of offloading all the codes to the fog device, the power consumption is given by,

$$P_{fog} = \sum_{C_k} P_{case2\_comk} + \sum_{C_k} P_{case2\_prok} + \sum_{C_k} P_{case2\_cmpk} + P_a \cdot T_{a2} \tag{34}$$

In case of offloading all the codes to the cloud server via the fog device, the power consumption is given by,

$$P_{cloud} = \sum_{C_k} P_{case3\_comk} + \sum_{C_k} P_{case3\_prok} + \sum_{C_k} P_{case3\_cmpk} + P_a \cdot T_{a2} \tag{35}$$

Let the number of requests arrived for offloading a full application to edge device, fog device and cloud server are r_1, r_2 and r_3, respectively. Then, the power consumption in the proposed strategy considering all the requests is given as,

$$P_{offload\_app} = \frac{r_1}{r} \cdot P_{edge} + \frac{r_2}{r} \cdot P_{fog} + \frac{r_3}{r} \cdot P_{cloud} \tag{36}$$

where r is the total number of requests given by $(r_1 + r_2 + r_3)$.

4.3.5 Power consumption in offloading partially an application program

In partial offloading of an application program the following cases are possible as we have considered in calculating delay:

Scenario 1 Some codes are executed inside the mobile device locally, and rest of the codes of that application program are executed inside the fog device.

Scenario 2 Some codes are executed inside the edge device, and rest of the codes of that application program are executed inside the fog device.

Scenario 3 Some codes are executed inside the fog device, and rest of the codes of that application program are executed inside the cloud server.

Power consumption in scenario 1

If a set of code C_{mob} execute locally inside the mobile device and another set of code C_{fog} execute inside the fog device, the power consumption is given by,

$$P_{partScenario1} = \sum_{C_k \in C_{mob}} P_{mi} \cdot (I_k/S_m) + \sum_{C_k \in C_{fog}} P_{case2\_comk} + \sum_{C_k \in C_{fog}} P_{case2\_prok} + \sum_{C_k \in C_{fog}} P_{case2\_cmpk} + P_a \cdot T_a \tag{37}$$

where T_a is the DIT access time if the mobile device itself can execute some of the codes of the application program.

Power consumption in scenario 2

If a set of code C_{edge} execute inside the edge device and another set of code C_{fog} execute inside the fog device, the power consumption is given by,

$$\begin{aligned} P_{partScenario2} = & \sum_{C_k \in C_{edge}} P_{case1\_comk} \\ & + \sum_{C_k \in C_{edge}} P_{case1\_prok} + \sum_{C_k \in C_{edge}} P_{case1\_cmpk} \\ & + \sum_{C_k \in C_{fog}} [P_{ts} \cdot (Da_{ukef}/D_u) + P_{tr} \cdot (Da_{dkef}/D_d)] \\ & + \sum_{C_k \in C_{fog}} P_{mi} \cdot (D_{ef}/S_{pro}) + \sum_{C_k \in C_{fog}} P_{mi} \cdot (I_k/S_f) + P_a \cdot T_{a1} \end{aligned} \tag{38}$$

where Da_{ukef}, Da_{dkef} are the amount of data transmission between edge and fog device in uplink and downlink, respectively, and D_{ef} is the distance between the edge and fog device.

Power consumption in scenario 3

If a set of code C_{fog} execute inside the fog device and another set of code C_{cloud} execute inside the cloud server, the power consumption is given by,

$$\begin{aligned} P_{partScenario5} = & \sum_{C_k \in C_{fog}} P_{case2\_comk} + \sum_{C_k \in C_{cloud}} P_{case3\_comk} \\ & + \sum_{C_k \in C_{fog}} P_{case2\_prok} \\ & + \sum_{C_k \in C_{cloud}} P_{case3\_prok} \\ & + \sum_{C_k \in C_{fog}} P_{case2\_cmpk} + \sum_{C_k \in C_{cloud}} P_{case3\_cmpk} + P_a \cdot T_{a2} \end{aligned} \tag{39}$$

If the number of requests arrived for offloading in scenarios 1, 2 and 3 are r_{s1}, r_{s2} and r_{s3}, respectively, the power consumption is given as,

$$P_{offload\_partapp} = \frac{r_{s1}}{r_s} \cdot P_{partScenario1} + \frac{r_{s2}}{r_s} \cdot P_{partScenario2} + \frac{r_{s3}}{r_s} \cdot P_{partScenario3} \tag{40}$$

where r_s is the total number of requests given by $(r_{s1} + r_{s2} + r_{s3})$.

4.4 Theoretical analysis

4.4.1 Delay

In this section we compare the delay in offloading an application using proposed C2OF2N and existing approach E2COM [9]. In E2COM [9] a mobile device $D_{1,}$ is registered under an access point offloads an application to another device D_2 registered under another access point. Here, D_2 is the edge device. If the device D_2 is unable, cloud server executes the application. If a mobile device offloads a code C_k to another device using E2COM [9], the total delay for communication in uplink and downlink is given by,

$$L_{exis\_comk} = (Da_{ukma}/D_u) + (Da_{ukaa}/D_u) + (Da_{ukae}/D_u) \\ (Da_{dkea}/D_d) + (Da_{dkaa}/D_d) + (Da_{dkam}/D_d) \tag{41}$$

where Da_{ukma}, Da_{dkam} are the amount of data transmission between mobile device and access point in uplink and downlink, respectively, Da_{ukaa}, Da_{dkaa} are the amount of data transmission between two access points in uplink and downlink, respectively, Da_{ukae}, Da_{dkea} are the amount of data transmission between access point and edge device, respectively.

The delay for propagation while offloading a code C_k to the edge device using E2COM [9] is given by,

$$L_{exis\_prok} = (D_{ma}/S_{pro}) + (D_{aa}/S_{pro}) + (D_{ae}/S_{pro}) \tag{42}$$

where D_{ma} is the distance between the mobile device and access point, D_{aa} is the distance between two access points, and D_{ae} is the distance between the access point and edge device.

The delay for executing a code C_k inside the edge device is given by,

$$L_{exis\_cmpk} = I_k/S_e \tag{43}$$

The total delay in offloading an application to an edge device using E2COM [9] is given by,

$$L_{exis} = \sum_{C_k} L_{exis\_comk} + \sum_{C_k} L_{exis\_prok} + \sum_{C_k} L_{exis\_cmpk} \tag{44}$$

In case of offloading to edge device in C2OF2N, the data transmission takes place between the mobile device and edge device. As both are registered under the same femtolet, directly the mobile and edge devices communicate with each other. But in E2COM [9] the data transmission takes place between mobile device and edge device via two access points because they are registered under two different access points. As a result the amount data transmission is less in C2OF2N if offloading occurs to

edge device. As the communication delay is directly proportional to the amount of data transmission, the communication delay is reduced in C2OF2N, which implies,

$$\sum_{C_k} L_{exis\_comk} > \sum_{C_k} L_{case1\_comk}.$$

Now as the mobile and edge devices communicate via two access points in E2COM [9], the propagation delay also increases than that of C2OF2N, where mobile and edge devices both register under the same femtolet and communicate directly. This implies,

$$\sum_{C_k} L_{exis\_prok} > \sum_{C_k} L_{case1\_prok}.$$

The execution delay is same because in C2OF2N and E2COM [9] both, the edge device executes the code. As communication and propagation delays are higher in E2COM [9] than C2OF2N, the total delay is reduced in case of offloading to edge device. In E2COM [9] if the edge device is unable, cloud server executes the code. But in C2OF2N the femtolet executes the code if edge device is unable. The communication and propagation delays between the mobile device and femtolet are much less than that of the mobile device and the cloud server. The femtolet has computation ability. Therefore, the delay in offloading a code to the femtolet is much less than the cloud server. As an application program is a set of codes, the delay in offloading an application is reduced, while femtolet is used instead of cloud server. Therefore, we conclude that the delay in offloading an application using C2OF2N is less than that of using E2COM [9]. In case of partial offloading in C2OF2N, if some codes are locally executed and rest of the codes are executed inside the femtolet, then the communication delay as well as propagation delay is reduced than offloading all the codes to the cloud server; accordingly the total delay decreases. If some codes are executed inside the edge device and rest of the codes are executed inside the femtolet, then also the communication delay as well as propagation delay is reduced than offloading all the codes to the cloud server; accordingly the total delay decreases. Hence, we can conclude that C2OF2N reduces the delay in offloading an application fully or partially.

Figure 6 shows the delays in three scenarios of application offloading using C2OF2N calculated using Eqs. (13), (14) and (15), respectively, with respect to the number of requests.

Figure 7 shows the delay in application offloading using our C2OF2N with respect to the total number of requests arrived. The result is compared to the delays in case of the existing multi-cloudlet-based offloading approaches [6–8] and cooperative offloading scheme E2COM [9]. Figure 7 shows that using C2OF2N the delay can be reduced by approximately 30, 22, 12 and 6% than the existing schemes [6–9], respectively. Offloading to edge device registered under the same femtolet reduces communication and propagation latencies than offloading to edge device under different access points [9] and offloading to cloudlets [6–8]; as a result the total delay is reduced in C2OF2N.

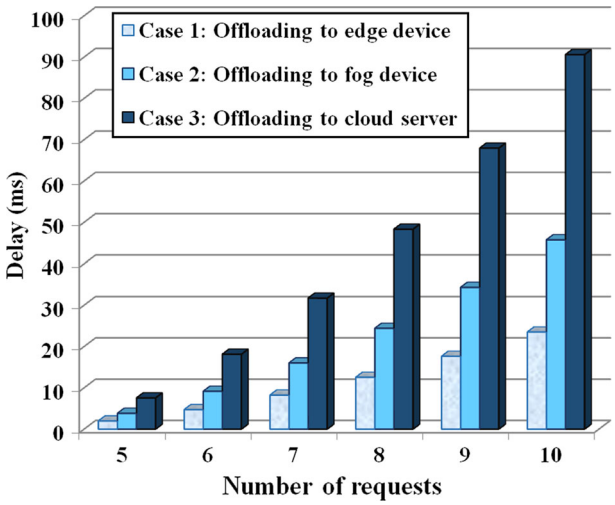

Fig. 6 Delay in three cases of C2OF2N

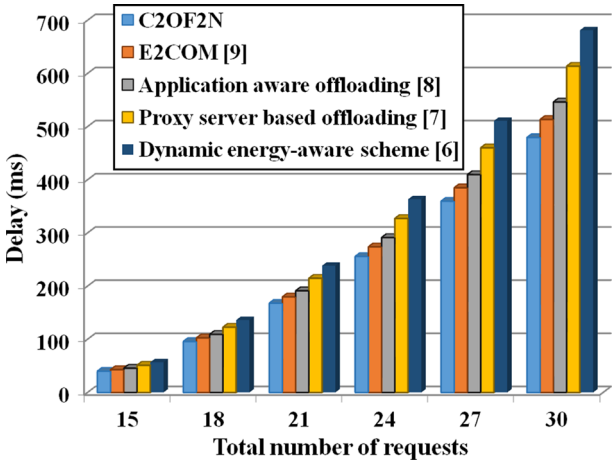

Fig. 7 Delay in C2OF2N and existing offloading schemes [6–9]

4.4.2 Power consumption

If a mobile device offloads a code C_k to another device using E2COM [9], the total power consumption during communication in uplink and downlink is given by,

$$P_{exis\_comk} = P_{ts} \cdot (Da_{ukma}/D_u) + P_{ts} \cdot (Da_{ukaa}/D_u) + P_{ts} \cdot (Da_{ukae}/D_u) \\ P_{tr} \cdot (Da_{dkea}/D_d) + P_{tr} \cdot (Da_{dkaa}/D_d) + P_{tr} \cdot (Da_{dkam}/D_d) \quad (45)$$

where Da_{ukma}, Da_{dkam} are the amount of data transmission between mobile device and access point in uplink and downlink, respectively, Da_{ukaa}, Da_{dkaa} are the amount of data transmission between two access points in uplink and downlink, respec-

tively, Da_{ukae}, Da_{dkea} are the amount of data transmission between access point and edge device, respectively.

The power consumption during propagation while offloading a code C_k to the edge device using E2COM [9] is given by,

$$P_{exis\_prok} = P_{mi} \cdot [(D_{ma}/S_{pro}) + (D_{aa}/S_{pro}) + (D_{ae}/S_{pro})] \tag{46}$$

where D_{ma} is the distance between the mobile device and access point, D_{aa} is the distance between two access points, and D_{ae} is the distance between the access point and edge device. The power consumption while executing a code C_k inside the edge device is given by,

$$P_{exis\_cmpk} = P_{mi} \cdot (I_k/S_e) \tag{47}$$

The total power consumption while offloading an application to an edge device using E2COM [9] is given by,

$$P_{exis} = \sum_{C_k} P_{exis\_comk} + \sum_{C_k} P_{exis\_prok} + \sum_{C_k} P_{exis\_cmpk} \tag{48}$$

In C2OF2N, the data transmission takes place between the mobile device and edge device while offloading an application. As the mobile and edge devices both register under the same femtolet, they communicate directly with each other. But in E2COM [9] the mobile and edge devices register under two different access points. Hence, the data transmission takes place between mobile device and edge devices via two access points. As a result the amount data transmission is less in C2OF2N than E2COM [9]. Therefore, the communication power is also less in C2OF2N than E2COM [9], which implies,

$$\sum_{C_k} P_{exis\_comk} > \sum_{C_k} P_{case1\_comk}.$$

As the mobile and edge devices communicate via two access points in E2COM [9], the power consumption during propagation also increases than that of C2OF2N. This implies,

$$\sum_{C_k} P_{exis\_prok} > \sum_{C_k} P_{case1\_prok}.$$

The power consumption during execution is same in C2OF2N and E2COM [9] because in both cases the edge device executes the code. As power consumption during communication and propagation is higher in E2COM [9] than C2OF2N, the total power consumption is reduced in case of offloading to edge device.

In E2COM [9] if the edge device is unable, cloud server executes the code. But in C2OF2N if edge device is unable, the femtolet executes the code. The power consumption during communication and propagation between the mobile device and femtolet

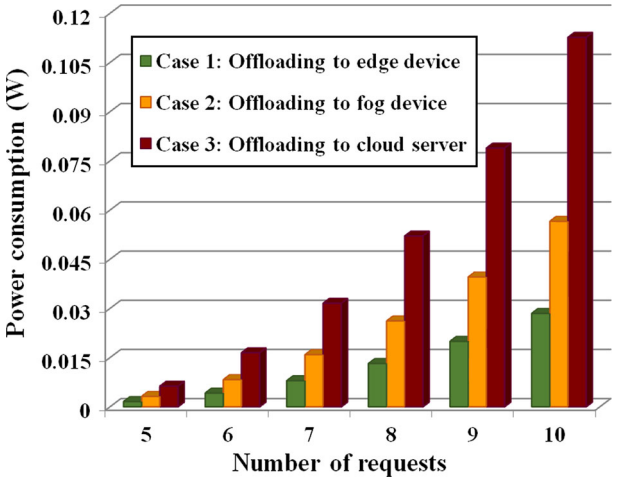

Fig. 8 Power consumption in three cases of C2OF2N

is much less than that of the mobile device and cloud server. The femtolet has computation ability. Thus, the power consumption in offloading a code to the femtolet is much less than the cloud server. As an application program is a set of codes, the power consumption in offloading an application is reduced, while femtolet is used instead of cloud server. Therefore, we conclude that the power consumption in offloading an application using C2OF2N is less than that of using E2COM [9]. In case of partial offloading in C2OF2N, if some codes are locally executed and rest of the codes are executed inside the femtolet, then the power involved during communication and propagation is reduced than offloading all the codes to the cloud server; accordingly, the total power consumption decreases. If some codes are executed inside the edge device and rest of the codes are executed inside the femtolet, then also the power involved in communication and propagation is reduced than offloading all the codes to the cloud server; accordingly, the total power consumption reduces. Hence, we can conclude that C2OF2N reduces the power consumption in offloading an application fully or partially.

Figure 8 shows the power consumptions in three scenarios of application offloading using C2OF2N calculated using Eqs. (33), (34) and (35), respectively, with respect to the number of requests.

Figure 9 shows the power consumptions in application offloading while using our C2OF2N with respect to the total number of requests arrived. The result is compared to the power consumptions in case of the existing multi-cloudlet-based offloading approaches [6–8] and cooperative offloading scheme E2COM [9].

Figure 9 shows that using C2OF2N the power consumptions are reduced by approximately 29, 21, 12 and 6% than the existing schemes [6–9], respectively. Offloading to edge device registered under the same femtolet reduces power consumptions during communication and propagation than offloading to edge device under different access points [9] and offloading to cloudlets [6–8]; as a result the total power consumption is reduced in C2OF2N.

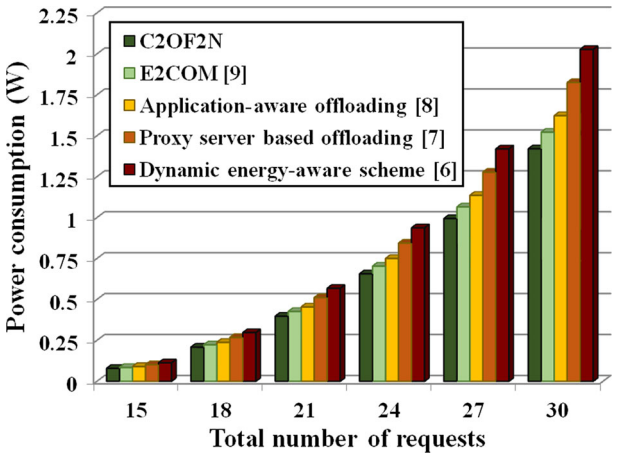

Fig. 9 Power consumption in C2OF2N and existing offloading schemes [6–9]

5 Performance evaluation

In this section first we have simulated C2OF2N in network simulator Qualnet version 7 where code offloading from mobile device to edge device and fog device is considered. Then, we have emulated code offloading from mobile device to edge device in the laboratory of West Bengal University of Technology (WBUT).

5.1 Simulation setup

The parameters used in simulation set up are given in Table 3.

The simulation scenario is presented in Fig. 10. In the figure nodes 1, 2, 3 and 4 are the mobile devices registered under node 5 femtolet. Here, nodes 1, 2, 3 and 4 are edge devices which form a cooperative network. Node 5 femtolet is the fog device. The fog device is connected with the cloud server denoted by node 6.

5.2 Simulation results

In this section throughput, carried load, average delay, average jitter and energy consumption of C2OF2N are determined.

5.2.1 Unicast received throughput

Figure 11 shows the average unicast received throughput in case of C2OF2N with respect to the amount of data transmission in the network. We have observed the unicast received throughput in three cases of application offloading using C2OF2N and calculated their average value. The average unicast received throughput is presented in Fig. 11. Figure 11 shows that the average unicast received throughput of C2OF2N

Table 3 Parameter values used in simulation

Layer	Parameter	Value
Physical layer	Radio type	802.11b
	Packet reception model	PHY 802.11b
	Antenna model	Omni directional
	Temperature	290.0 K
	Noise factor	10.0
MAC layer	Used protocol in MAC layer	802.11
Network layer	Used protocol in Network layer	IPV4
	Protocol for routing	Bellman Ford
Transport layer	Buffer size	16384 bytes
Battery model	Battery model	Linear model
	Full capacity of battery	1200 mA h
Scenario properties	Time of simulation	300 s
CBR properties	Size of item	512 bytes, 1024 bytes, 2048 bytes
	Number of items sent	100

is approximately 20,000–1,60,000 bits/s, respectively, for 50–250 KB of data transmission in the network.

5.2.2 Carried load

Figure 12 shows the carried load in case of C2OF2N with respect to the amount of data transmission in the network. We have observed the carried load in three cases of application offloading using C2OF2N and calculated their average value. The average carried load is presented in Fig. 12. Figure 12 shows that the average of C2OF2N is approximately 2000–8000 bits/s, respectively, for 50–250 KB of data transmission in the network.

In theoretical analysis we have already discussed that using proposed C2OF2N the power consumption and delay are reduced than the cloudlet-based approaches [6–8]. In this section we compare the delay, jitter and energy consumption in our C2OF2N with existing cooperative offloading scheme E2COM [9].

5.2.3 Average delay

The average delays in three different cases of application offloading in C2OF2N are observed, and their average value is calculated. Figure 13 shows the average delay in case of C2OF2N and E2COM [9] with respect to the amount of data transmission in the network. Figure 13 shows that the average delay of C2OF2N and E2COM [9] is approximately 0.01–0.06 and 0.02–0.07 s, respectively, for 50–250 KB of data transmission in the network. Hence, our C2OF2N reduces the average delay up to 12% approximately than E2COM [9].

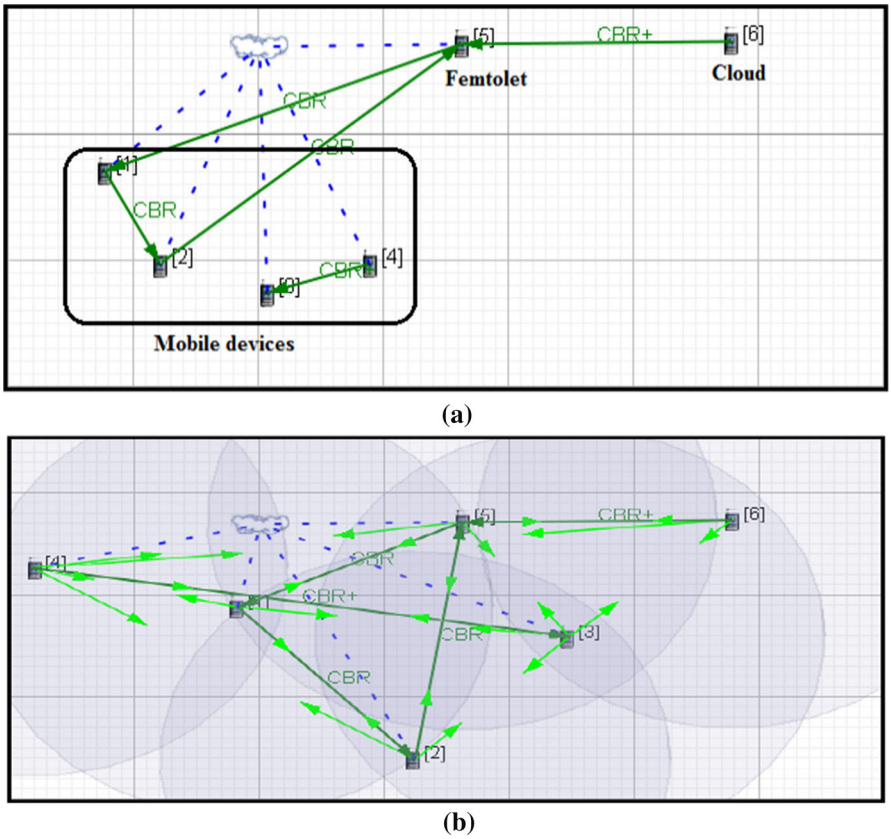

Fig. 10 Simulation scenario of C2OF2N. **a** Created simulation scenario before start up of experiment, **b** simulation scenario during progress of experiment

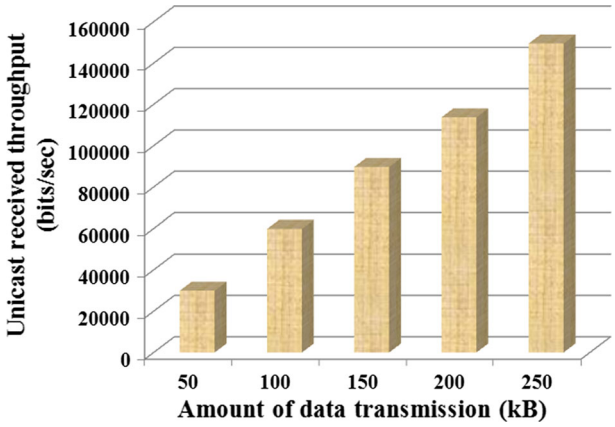

Fig. 11 Average unicast received throughput (bits/s) of C2OF2N

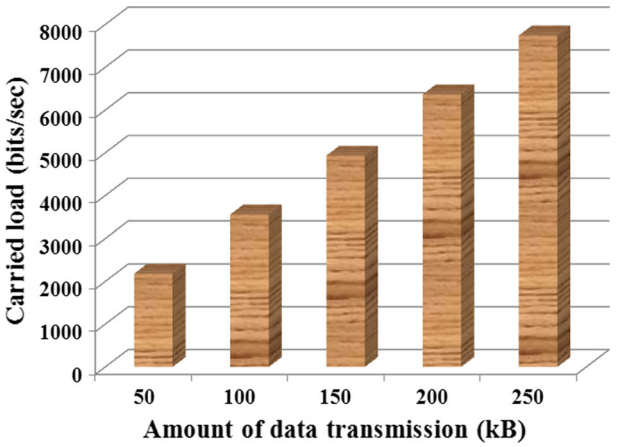

Fig. 12 Average carried load (bits/s) of C2OF2N

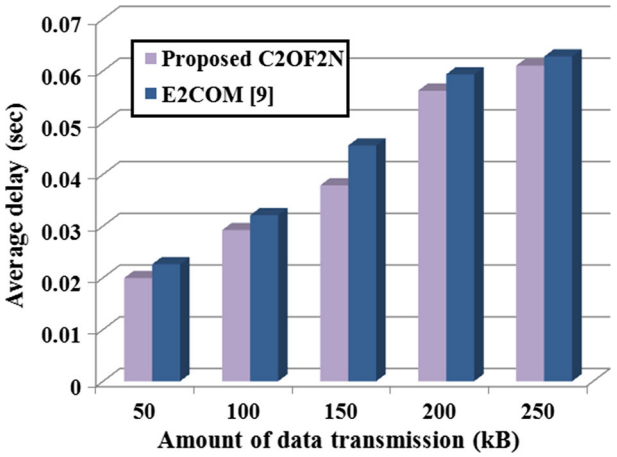

Fig. 13 Comparison of average delay (s) between C2OF2N and E2COM [9]

5.2.4 Average jitter

The average jitters in three different cases of application offloading in C2OF2N are observed, and their average value is calculated. Figure 14 shows the average jitter in case of C2OF2N and E2COM [9] with respect to the amount of data transmission in the network. Figure 14 shows that the average jitter of C2OF2N and E2COM [9] is approximately 0.01–0.06 and 0.01–0.07 s, respectively, for 50–250 KB of data transmission in the network. Hence, our C2OF2N reduces the average jitter up to 18% approximately than E2COM [9].

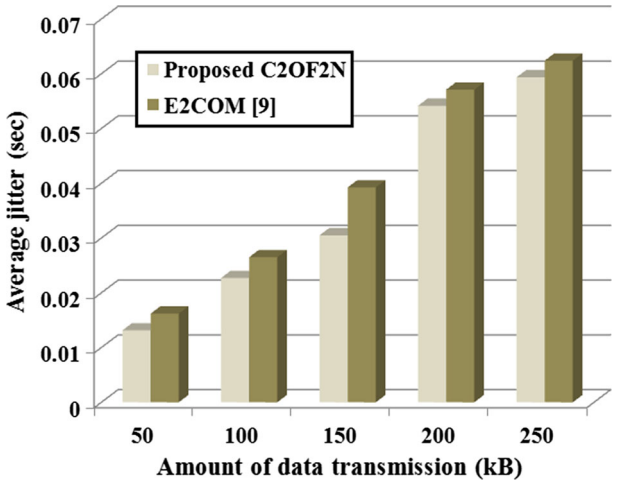

Fig. 14 Comparison of average jitter (s) between C2OF2N and E2COM [9]

5.2.5 Energy consumption

The energy consumption in transmit, receive and idle modes are summed up to determine the total energy consumption. The total energy consumptions in three different cases of application offloading in C2OF2N are observed, and their average value is calculated. Figure 15 shows the energy consumption in case of C2OF2N and E2COM [9] with respect to the amount of data transmission in the network. Figure 15 shows that the energy consumption in C2OF2N and E2COM [9] is approximately < 2.5 mW h and ≤ 3 mW h, respectively, for 50–250 KB of data transmission in the network. Hence, our C2OF2N reduces the energy consumption by approximately 15% than E2COM [9].

5.3 Experimental analysis of code offloading to edge device

The proposed strategy is experimentally implemented in the MCC laboratory of WBUT. The configurations of the mobile devices used in our experiment along with their executable application types are given in Table 4.

The mobile devices are connected to the wireless local area network (WLAN) which is controlled by CISCO WLC 2504. A femtolet can serve at most 32 users. We have considered three types of applications, and network speed is 1 Gbps. M_1, M_2 and M_3 register under a femtolet and collaborate their resources. These three mobile devices form a group $\{M_1, M_2, M_3\}$. As we observe from Table 4, M_1, M_2 and M_3 executes matrix operation-related applications, recursive function-related applications and sorting- and searching-related applications, respectively. When M_1 has to offload a code of recursive function-related applications, it will form cooperation with M_2. When M_1 has to offload a code of sorting- and searching-related applications, it will form cooperation with M_3. When M_2 has to offload a code of matrix operation-related

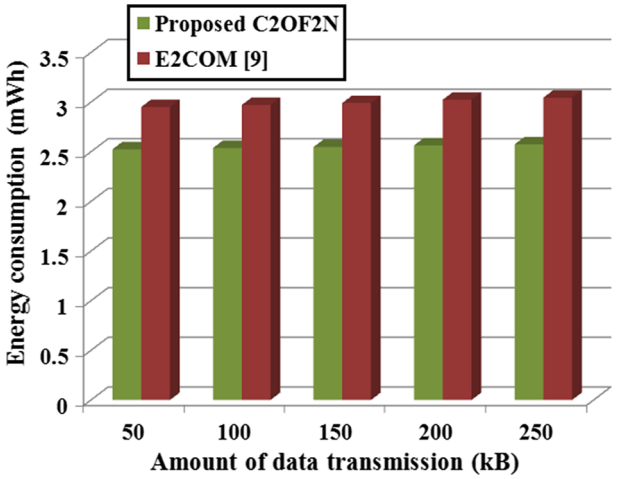

Fig. 15 Comparison of energy consumption (mW h) between C2OF2N and E2COM [9]

Table 4 Configurations of mobile devices used in experiment

Mobile device	RAM (GB)	Storage/HDD (GB)	Processor	Executable application
M_1:Asus ZenFone Max	2	16	Intel Atom Z2560 1.6 GHz	Matrix operation-related applications
M_2:Samsung smart phone Galaxy J2	2	8	1.3 GHz quad-core Exynos 3475	Recursive function-related applications
M_3:Lenevo laptop	2	320	Intel(R) Pentium(R) CPU B940 @ 2.00 GHz	Sorting- and searching-related applications

applications, it will form cooperation with M_1. When M_2 has to offload a code of sorting- and searching-related applications, it will form cooperation with M_3. When M_3 has to offload a code of matrix operation-related applications, it will form cooperation with M_1. When M_3 has to offload a code of recursive function-related applications, it will form cooperation with M_2.

Table 5 shows the time and power consumptions in offloading different codes of different applications using our proposed approach. The total time consumption is calculated as the time of receiving result minus the time of sending request to the edge device. The computational time of each task is shown in Table 5. The power is calculated as: (the power consumption by the requesting device per time unit in idle mode * computational time) + (the power consumption by the requesting device per time unit for sending data* time of sending code to the edge device) + (the power consumption by the requesting device per time unit for receiving data* time of receiving result from the edge device).

Table 5 Time and power consumption in code offloading using C2OF2N

Requesting application	Requesting code	Mobile device	Edge device	Computational time (s)	Total time (s)	Total power (W)	Remarks
Recursive function-related applications	Tower of Hanoi for number of disk=3	M_1	M_2	1.756	1.762	0.21	M_2 executes recursive function-related codes. Thus, the requested codes are offloaded to M_2 based on cooperation
	Factorial of 10	M_3		2.892	2.899	0.35	
Matrix operation-related applications	Matrix determinant for matrix of order 10×10	M_2	M_1	2.232	2.252	0.27	M_1 executes matrix operation-related codes. Thus, the requested codes are offloaded to M_1 based on cooperation
	Matrix addition of order 30×40	M_3		2.74	2.747	0.33	
Sorting- and searching-related applications	Insertion sort	M_1	M_3	30.615	30.75	3.69	M_3 executes sorting- and searching-related codes. Thus, the requested codes are offloaded to M_3 based on cooperation
	Linear search	M_2		28.513	28.67	3.44	

M_1 and M_3 have to offload codes of Tower of Hanoi and factorial calculation, respectively. Both of these codes contain recursive functions. Hence, these codes are offloaded inside M_2 which is for executing recursive function-related applications. There exists cooperation between M_1 and M_2, and between M_3 and M_2.

M_2 and M_3 have to offload codes of calculating matrix determinant and matrix addition, respectively. Both of these codes contain matrix-related operations. Hence, these codes are offloaded inside M_1 which is for executing matrix operation-related applications. There exists cooperation between M_2 and M_1, and between M_3 and M_1.

M_1 and M_2 have to offload codes of insertion sort and linear search, respectively. Both of these codes contain sorting- and searching operations. Hence, these codes are offloaded inside M_3 which is for executing sorting- and searching-related applications. There exists cooperation between M_1 and M_3, and between M_2 and M_3.

In C2OF2N offloading occurs from a mobile device to another mobile device (edge device); the communication time and communication power consumptions are much less. As a result the time and power consumptions are reduced. By reducing the time and power consumptions faster and power-efficient service can be provided.

5.4 Contribution of C2OF2N with respect to existing schemes

The novelty and contributions of C2OF2N with respect to the existing offloading strategies are presented in Table 6.

It is observed that the proposed approach is novel and reduces power and delay than the existing strategies.

6 Conclusion and future work

We have proposed a cooperative code offloading strategy for femtolet-based fog network. Mobile devices register under a femtolet which is an indoor base station having large internal storage and the ability of computation. The femtolet serves as a fog device. The mobile devices connected with the same femtolet form a group where each mobile device can offload a particular type of application. When a mobile device in the group has to offload a code, it requests another mobile device from the group based on the application type. The executing device is the edge device. If the edge device is unable, then the code is executed inside the fog device femtolet. The power and delay involved in our method are calculated and compared with the existing offloading schemes. This is observed that the proposed scheme reduces the delay in offloading by approximately 22 and 12% than the existing proxy server-based and application-based code offloading strategies. We also observe that the proposed method diminishes the power consumption in offloading by approximately 21 and 12% than the existing proxy server-based and applicatio-based code offloading strategies. The proposed scheme is experimentally emulated, and the time and power consumption are determined. The proposed approach is simulated in Qualnet 7, and the average delay, jitter, carried load, unicast received throughput and energy consumption are determined. The simulation results illustrate that our strategy reduces average delay, jitter and energy consumption by up to 12, 18 and 15%, respectively, than the existing

Table 6 Comparison of proposed and existing offloading schemes

Features	Energy aware cloudlet allocation in multi-cloudlet scenario [6]	Proxy server-based offloading in multi-cloudlet scenario [7]	Application aware offloading in multi-cloudlet scenario [8]	E2COM [9]	Proposed C2OF2N
Working model	Cloudlet providing energy efficiency is selected for offloading	Offloading takes place from mobile device to the cloudlet selected by a proxy server	Offloading takes place from mobile device to the cloudlet based on the type of application to be offloaded	Cooperative offloading model is proposed. Here, one mobile device offloads its task to another mobile device belonging under the coverage of another access point. If the mobile device is unable, then offloading takes place to the cloud	Based on the type of application, offloading takes place from mobile device to another mobile device (edge device) both registered under the same femtolet(fog device) providing communication and computation facilities
Offloading occurs inside	Cloudlet	Cloudlet	Cloudlet	Mobile device and cloud	Edge device and fog device
Femtolet is used	×	×	×	×	✓
Fog computing is used	×	×	×	×	✓
Application aware offloading occurs	×	×	✓	×	✓
Cooperative federation is used	×	×	×	✓	✓

Table 6 continued

Features	Energy aware cloudlet allocation in multi-cloudlet scenario [6]	Proxy server-based offloading in multi-cloudlet scenario [7]	Application aware offloading in multi-cloudlet scenario [8]	E2COM [9]	Proposed C2OF2N
Reduction in delay in proposed scheme than [6–9]	30%	22%	12%	6%	Not applicable
Reduction in power than [6–9]	29%	21%	12%	6%	Not applicable
Remarks	Proposed C2OF2N is novel and reduces power consumption and delay than the existing schemes.				

cooperative offloading method. Therefore, the proposed approach is recommended for low power and fast offloading in femtolet-based indoor mobile cloud network.

In our work the femtolet offloads the storage and sometimes computation. But the femtolet may be damaged. Then, how the cooperative offloading will occur in a seamless manner between the mobile devices is a challenging issue. It may also happen that the femtolet is unable to provide service due to technical fault. In such a case offloading to any nearby cloudlet or the remote cloud server in an energy-efficient way is another future research challenge. The mobile devices are served by the femtolet, and the femtolet is connected with the remote cloud server. Thus, seamless cloud service provisioning with femtolet, nearby cloudlets and remote cloud server in a power optimized way is a promising future research scope of this work. As the proposed approach is based on femtolet, it is applicable for indoor region only. For the outdoor region, outdoor base stations will be required to be incorporated, where the offloading will occur either inside the cloudlet or the cloud server. Hence, the extension of our approach over the outdoor environment is also a promising research scope.

Acknowledgements We are grateful to Department of Science and Technology (DST) project funding SR/FST/ETI-296/2011 and TEQIP III. This work is partially supported by ARC Future Fellowship and Melbourne-Chindia Cloud Computing Research Network.

References

1. Mukherjee A, De D (2016) Femtolet: a novel fifth generation network device for green mobile cloud computing. Simul Model Pract Theory 62:68–87
2. Mukherjee A, Bhattacherjee S, Pal S, De D (2013) Femtocell based green power consumption methods for mobile network. Comput Netw 57:162–178
3. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. Pervasive Comput 8:14–23
4. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Future Gener Comput Syst 29:84–106
5. Verbelen T, Pieter S, Filip D T, Bart D (2012) Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services, ACM, pp 29–36
6. Gai K et al (2016) Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. J Netw Comput Appl 59:46–54
7. Mukherjee A, De D, Roy DG (2016) A power and latency aware cloudlet selection strategy for multi-cloudlet environment. IEEE Trans Cloud Comput 99:1–1
8. Roy DG, De D, Mukherjee A, Buyya R (2016) Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. J Supercomput 73:1672–1690
9. Song J, Cui Y, Li M, Qiu J, Buyya R (2014) Energy-traffic tradeoff cooperative offloading for mobile cloud computing. In: IEEE 22nd International Symposium of Quality of Service, pp. 284–289
10. TawalbehL A, JararwehY DosariF (2015) Large scale cloudlets deployment for efficient mobile cloud computing. J Netw 10:70–76
11. MahmudR, BuyyaR (2017) Fog computing: a taxonomy, survey and future directions. In: DiMartino B, Li K, Yang L, Esposito A (eds) Internet of everything: algorithms, methodologies, technologies and perspectives. Springer, Singapore, pp. 103–130. ISBN 978-981-10-5860-8
12. Jalali F et al (2016) Fog computing may help to save energy in cloud computing. IEEE J Sel Areas Commun 34:1728–1739
13. Ahmad M, Amin MB, Hussain S, Kang BH, Cheong T, Lee S (2016) Health fog: a novel framework for health and wellness applications. J Supercomput 72:3677–3695
14. Shuja J, Mustafa S, Ahmad RW, Madani SA, Gani A, Khan MK (2017) Analysis of vector code offloading framework in heterogeneous cloud and edge architectures. IEEE Access 5:24542–24554

15. Durao F, Carvalho JFS, Fonseka A, Garcia VC (2014) A systematic review on cloud computing. J Supercomput 68:1321–1346
16. Wang X, Wang J, Wang X, Chen X (2017) Energy and delay tradeoff for application offloading in mobile cloud computing. IEEE Syst J 11:858–867
17. Singh S, Chana I (2015) QRSF: QoS-aware resource scheduling framework in cloud computing. J Supercomput 71:241–292
18. Chunlin L, LaYuan L (2015) Cost and energy aware service provisioning for mobile client in cloud computing environment. J Supercomput 71:1196–1223
19. 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
20. Shuja J, Gani A, ur Rehman MH, Ahmed E, Madani SA, Khan MK, Ko K (2016) Towards native code offloading based MCC frameworks for multimedia applications: a survey. J Netw Comput Appl 75:335–354
21. Chen X, Chen S, Zeng X, Zheng X, Zhang Y, Rong C (2017) Framework for context-aware computation offloading in mobile cloud computing. J Cloud Comput 6:1
22. Ahmed E, Rehmani MH (2016) Mobile edge computing: opportunities, solutions, and challenges. Future Gener Comput Syst 70:59–63
23. Roman R, Lopez J, Mambo M, Mobile Edge Computing, Fog et al. (2016) A survey and analysis of security threats and challenges, arXiv preprint arXiv:1602.00484
24. Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34:3590–3605
25. Chen X, Jiao L, Li W, Fu X (2016) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw 24:2795–2808