


RESEARCH

Open Access



A strategy for joint service offloading and scheduling in heterogeneous cloud radio access networks

Olfa Chabbouh^{1*} , Sonia Ben Rejeb¹, Zied Choukair¹ and Nazim Agoulmine²

Abstract

Cloud radio access network is one of the most promising cellular networks for the next generation of mobile networks. The basic idea of cloud RAN (radio access network) is virtualizing and centralizing the intelligent part of the base station, the base band unit, and keeping remote radio heads on cell site enabling a centralized processing and management. Offloading data computation to edge cloud was proposed as a solution to deal with resource limitation while keeping a good quality of service. In this paper, we propose a strategy to jointly handle offloading decision and offloading request scheduling in cloud RAN. We aim to improve network quality of service while reducing the scheduling cost expressed in terms of overload, network delay, and migration cost. Numerical results show that the proposed approach is able to reduce the response time of the applications, mobile terminal energy consumption, and total execution cost.

Keywords: Cloud RAN, 5G networks, HetNets, Offloading, Scheduling, QoS

1 Introduction

With the success of smartphones, mobile developers are becoming more ingenious in creating sophisticated applications to attract users such as face recognition, interactive gaming, and augmented reality applications. However, these applications are resource-intensive, i.e., they require high processing and energy capabilities to be executed. Finding a tradeoff between limited resources and battery lifetime of mobile devices and resource-intensive applications is a big challenge for next-generation mobile platform development [1].

Mobile cloud computing was proposed as a solution to tackle this challenge [2]. Offloading of total or part of application workflow to a resource-rich cloud infrastructure helps to increase mobile devices capabilities. However, computation offloading to remote central cloud will not solve the problem while mobile users may experience long latency for data exchange with the central cloud through the wide area network. Since it is very hard to reduce the latency in a wide area network,

mobile cloud computing based cloudlets was proposed as a solution [3]. The basic idea consists on leveraging the physical proximity by offloading computation to servers via a Wi-Fi access point. However, due to limited coverage of Wi-Fi access points, services cannot be provided everywhere. Besides, cloudlets are based on servers with small or medium resources which may not satisfy the QoS (quality of service) of a large number of users.

In order to meet these challenges, mobile edge cloud computing was proposed as an innovative mobile cloud computing paradigm which complements the cloudlet concept [4]. The concept of mobile edge cloud computing is to provide cloud computing resources at the edge of radio access networks close to mobile end users. Using this infrastructure will allow to reduce latency by deploying a fiber transport network between base stations and edge cloud data centers. Endowing base stations with additional computation and storage resources is expected to enhance mobile users' QoS anywhere and at any time [5].

In a C-RAN (cloud radio access network) infrastructure, base band units are moved from cell site to a central data center. With this approach, all the radio access network functionalities are centralized in the cloud.

* Correspondence: olfa.chabbouh@supcom.tn

¹High School of Communication of Tunis (Sup'com), Carthage University, Ariana, Tunisia

Full list of author information is available at the end of the article

Centralized processing enables indeed more advanced and efficient network coordination and management. In the cell site, RRH (radio remote head) is still responsible for transmitting radio signal, amplification of signal power and analog to/from digital conversion, while all the base band signal processing parts, including physical, mac, and upper layers, which require higher processing resources, are relocated from the cell site to a centralized BBU (base band unit) in the operator cloud infrastructure. The interface between the numerous RRH and BBU is named CPRI (common public radio interface). This interface supports a bidirectional constant bit rate protocol that requires accurate synchronization and strict latency control. Other protocols have also been proposed for this interface such as OBSAI (Open Base Station Architecture Initiative) and ORI (open radio equipment interface). The general C-RAN architecture is illustrated in Fig. 1.

In this paper, we propose a novel strategy for data offloading in cloud RAN-based mobile edge cloud computing. We aim to fully leverage the potential of such infrastructure throughout joint offloading decision and offloading requests scheduling in the edge cloud. One of the most critical issues impacting data offloading performances is task scheduling in cloud resources.

For example, when too many applications are offloaded to the same edge cloud, if all of them are executed on the same container, it will be highly overloaded. In return, this situation will lead to high energy consumption and long application response time. Therefore, in order to efficiently benefit from data computation offloading, we need to address two key challenges:

- (1) How to choose between local execution on mobile device and offloading to the cloud?

- (2) Once an application is offloaded to the cloud, how to schedule it among the available resources?

The rest of the paper is organized as follows: In the next section, different related works are discussed. In Section 3, we explain our strategy for joint offloading decision and offloading request scheduling. Then, in Section 4, we study the system performances and present simulation results. Finally, Section 5 concludes this paper.

2 Related work

Internet data traffic is increasing exponentially, especially the portion of traffic going through mobile networks. That is why mobile data computation offloading has become an important issue in cellular networks. As a result, various cloud offloading systems were proposed in the literature. MAUI (Mobile Assistance Using Infrastructure) [1] and ThinkAir [6] describe hardware components and propose to offload data in order to optimize energy consumption of mobile devices. However, they ignore other aspects of offloading. CloneCloud [7] proposes to improve application partitioning between the device and the cloud with the purpose of reducing energy consumption or execution time.

In addition to mobile cloud frameworks, many other research works have focused on offloading decision-making issue in MCC (mobile cloud computing) based systems. However, most of them have focused on the issue of energy consumption without considering other parameters that can affect the offloading process. For example, authors in [8] proposed an offloading decision mechanism based on computation, communication, and compilation energies comparison. The main objective of the proposed process is conserving energy on mobile terminal. It consists in comparing different energy consumption values of different execution strategies and choosing the alternative which have the lowest energy cost. In [9], Liu et al. proposed an offloading decision algorithm based on application deadline and communication quality constraints. The proposed offloading decision process helped the cloud controller to choose tasks for offloading in order to minimize mobile handset energy consumption. In [10], authors proposed CADA (context-aware offloading decision algorithm) in order to offload to the cloud servers. CADA uses a profiler containing the location of the mobile user and time-of-day in order to make the mobile offloading decisions. However, this method generates a lot of overhead and requires a lot of memory in order to store users' profiles. In [11], authors have proposed an energy-aware data offloading scheme for cloud RAN. The centralized BBU makes offloading decision considering the mobile devices transmission rate and energy consumption of both cellular and Wi-Fi networks. It benefits from the

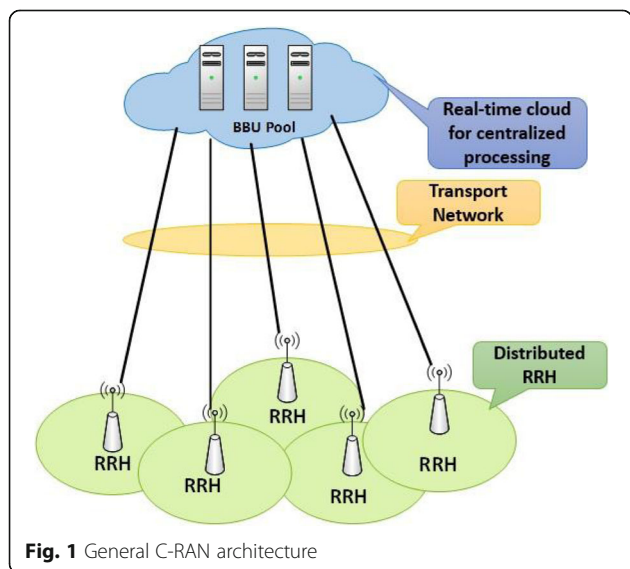


Fig. 1 General C-RAN architecture

centralized characteristic of C-RAN in order to schedule mobile terminals' computation offloading from RRH to Wi-Fi access point. In [12], authors proposed to jointly optimize communication resources (downlink and uplink beamforming), computation resources (power and computation capability allocation), and offloading decision in cloud RAN to minimize the network energy consumption while satisfying applications' delay requirements.

There is also a significant body of literature on task scheduling in the cloud mostly focusing on traditional cloud systems. Authors in [13] have proposed a task scheduling scheme to achieve the cooperation between local cloud and the Internet cloud. Applications are firstly classified according to their delay requirements before being served. However, with such an approach, most end users have to wait before being served and QoS decreases dramatically when the arrival rate increases. For this reason, authors have then designed a threshold-based policy to cooperatively schedule the local cloud and the Internet cloud. The objective is to maximize the probability that tasks can be executed within their delay requirements. In [14], a job scheduling scheme in the cloud computing clusters considering job resource requirements and completion time sensitivity is proposed. The problem is formulated as a maximization of the minimum utility achieved across all the jobs in the cluster, where the job utilities are functions of their completion times. In [15], authors have presented an online scheduling scheme that aims to minimize the average task queuing delay while accounting for task execution times. As a first step, upon the arrival of a new task, the scheduler tries to find an available server for assigning and executing this recently submitted task. If the scheduler does not find any server for the task, it will be putted in the queue. Then, the scheduler tries to find the task with the shortest execution time among the tasks that are already in the queue and fits it on the recently released server. In another work, authors have proposed in [16] a task scheduling mechanism which takes care of deadline and cost. Based on the concept of space-shared scheduling policy, this work presents a CDB (cost-deadline based) task scheduling algorithm to schedule tasks by taking into account task penalty and provider profit. Simulations show that if the number of virtual machines and datacenters decreases with the decrease of the number of cloudlets, the proposed algorithm misses deadline. Cost-based scheduling using linear programming was also investigated in [17]. Authors proposed SAH-DB (a task scheduling algorithm based on delay-bound constraint) in order to improve the task execution concurrency: when a task is received, all the resources (CPU, memory, and network) are sorted in a descending order based on the resources processing capacity, then the task is dispatched to resources with the minimum execution time.

To summarize our state-of-the-art analysis, we can state that existing contributions in the area of mobile computation offloading optimization have mainly focused on the energy consumption and latency. In the area of task scheduling, in which the main focus was on the jobs' completion time, we propose in this work a scheduling optimization mechanism that aims to reduce the cost of task scheduling. Unlike previous works, we model the cost of tasks as a function of overloading, network delay, and migration. The proposed resource management strategy takes mainly into account the available resources, resource requirements, deadlines, and load balancing in Cloud-RRH. These two problems are addressed separately and there are very few contributing addressing these two aspects in a holistic way. In this work, we propose to address the two problems at the same time proposing a joint optimization of offloading decision and application scheduling in the edge cloud which from our viewpoint is necessary. In our previous contribution [18], we have proposed a dynamic multi-parameter offloading decision scheme in order to adapt offloading decision to the current network state. In this work, we extend significantly this contribution proposing a global offloading strategy which combines offloading decision and task scheduling optimization. The originality of our algorithm is that it takes into consideration in the decision-making several parameters related to the network state, the MT mobility, and capabilities, as well as the tasks to offload.

3 Computation offloading and task scheduling in H-CRAN

3.1 System model

3.1.1 Computation offloading system model

The scenario is depicted in Fig. 2. We consider a H-CRAN (heterogeneous cloud radio access network) composed of H-RRHs (high remote radio heads) that act as macro-cells and L-RRHs (low remote radio heads) that act as small cells. In our previous works, we proposed to add an edge cloud, the Cloud-RRH [19]. It represents additional cloud resources close to the mobile end user. While in traditional cloud RAN architecture all RAN functionalities are centralized in the cloud, we proposed to flexibly split RAN functionalities. Besides, the Cloud-RRH contains additional computation and storage resources for data offloading. We also make the following assumptions: (i) mobile applications that are utilized by the mobile user are installed on the mobile device, on the cloud server, and also on the Cloud-RRH; (ii) even if the interface between mobile terminal and clouds may provide different rate and delay values, mobile broadband connectivity does not change during the application processing time. Note that the second assumption means that we considered applications with no large

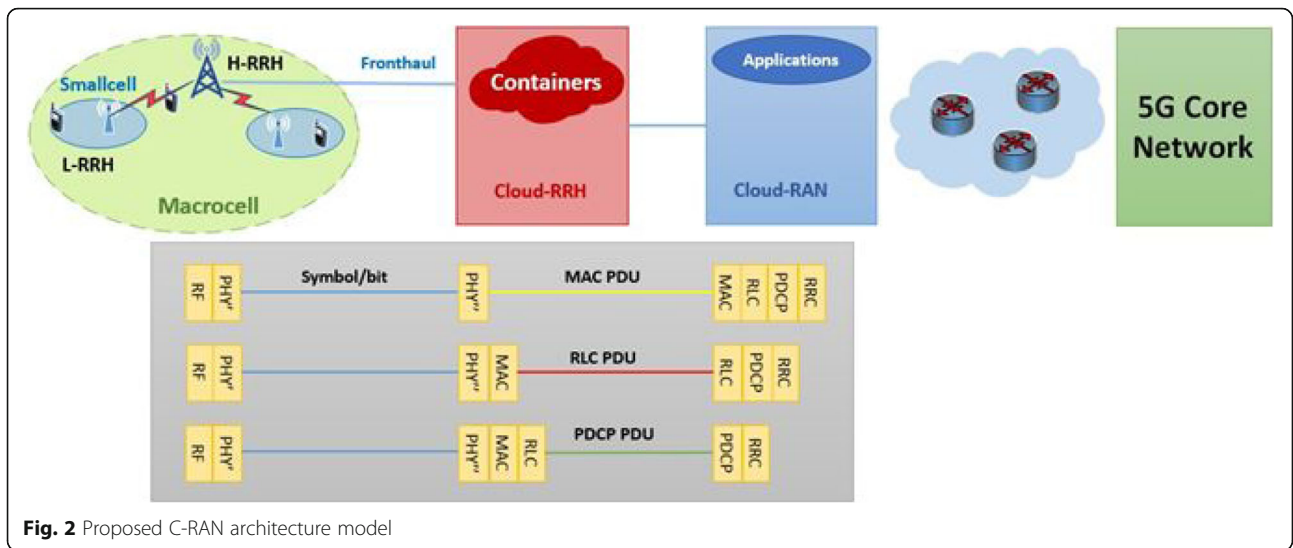


Fig. 2 Proposed C-RAN architecture model

processing time. This condition has also been assumed by most of the related works [5, 20–23].

The system is composed of M mobile users that can be served by either a high or a low RRH. We consider uplink. The time needed to transfer S_{up} bits in the UL (uplink) connection between mobile device and the serving RRH t_{up} depends only on the uplink data rate r_{up} and the number of bits to be transmitted, i.e., $t_{up} = S_{up}/r_{up}$. Similarly, for DL (downlink) transmission, after remote processing, from Cloud-RRH to the serving RRH, the time required is $t_{dl} = S_{dl}/r_{dl}$, with S_{dl} the number of bits to transmit and r_{dl} the downlink data rate.

The measurements provided in [22] proved that the power consumed by the mobile device in UL increases with the uplink transmission power, p_{tx} , while a baseline power is consumed just for having the transmission chain switched on, whereas the power consumed in DL increases with the downlink data rate, r_{dl} , and a baseline power is consumed just for having the reception chain switched on. Based on these results, we adopted the following models of power consumption at the mobile device in both UL and DL:

$$p_{ul} = k_{(tx,1)} + k_{(tx,2)}p_{tx} \tag{1}$$

$$p_{dl} = k_{(rx,1)} + k_{(rx,2)}r_{dl} \tag{2}$$

where $k_{(tx,1)}$, $k_{(tx,2)}$, $k_{(rx,1)}$, and $k_{(rx,2)}$ are constants.

The maximum rate supported by the channel with M users depends on the quality of the channel and the transmission power. It is given using Shannon's theorem by the following expressions in UL and DL:

$$r_{(up,m)} = B \log \left(1 + G_{up}p_{(tx,m)} \right) \tag{3}$$

$$r_{(dl,m)} = B \log \left(1 + G_{dl}p_{(tx,RRH)} \right) \tag{4}$$

where G_{up} and G_{dl} are the channel gain normalized by the average power of the noise and interference over the bandwidth, respectively, in uplink and downlink, $p_{(tx,m)}$ and $p_{(tx,RRH)}$ represent the transmission power of the mobile user and RRH, respectively, and B is the channel bandwidth.

According to (1) and (2), the energy spent by mobile device in UP and DL is given by the following equations:

$$E_{up} = k_{(tx,1)}t_{up} + k_{(tx,2)}t_{up}p_{tx} \tag{5}$$

$$E_{dl} = k_{(rx,1)}t_{dl} + k_{(rx,2)}t_{dl}r_{dl} \tag{6}$$

Using (3), we can express p_{tx} as $p_{tx} = \frac{r_{up}}{2 \frac{B}{G_{up}} - 1}$. Therefore, the energy consumed by the mobile device for offloading is given by the following equation:

$$\begin{aligned} E_{off} &= E_{up} + E_{dl} \\ &= k_{(tx,1)}t_{up} + k_{(tx,2)}t_{up} \frac{2 \frac{S_{up}}{t_{up}} \cdot B}{G_{up}} - 1 + k_{(rx,1)}t_{dl} + k_{(rx,2)}S_{dl} \end{aligned} \tag{7}$$

The energy spent by the mobile device in the local processing is considered to be proportional to the number of processed bits. It is given by the following equation:

$$E_{loc} = \epsilon_0 S \tag{8}$$

where ϵ_0 is a constant that accounts jointly for the Joules/cycle and cycles/bit at the mobile device processor and S is the total number of bits.

Concerning the latency, we considered t_0 as the time needed to process one bit at the mobile device and t_1 as the time needed to process one bit at the RRH.

Therefore, the time needed for local execution is given by the multiplication of t_0 by the number of bits. Whereas if the execution is offloaded to the Cloud-RRH, the time required for processing is given by the sum of the time required to transfer the bits from the mobile device to the serving RRH through the UL transport network, the time for the remote cloud to execute the offloaded computation, and the time to transfer all the output bits through the DL. Latency for both local processing and offloading are expressed by the following equations:

$$L_{loc} = t_0 S \quad (9)$$

$$L_{off} = t_{up} + t_1 S + t_{dl} \quad (10)$$

3.1.2 Task scheduling system model

We assume that N predefined containers are running on each Cloud-RRH and each container is characterized by its available capacity resources CPU $_i$, RAM $_i$, and Net $_i$, $i \in N$. Each offloading request is composed of T tasks that have to be executed with a deadline D , and each task is characterized by its CPU $_j$, RAM $_j$, and Net $_j$ and has an expected execution time Tex_j , $j \in T$. We consider a binary variable $t_{(i,j)}$ to indicate if a task j is allocated to a container i or not:

$$t_{(i,j)} = \begin{cases} 1 & \text{if task } j \text{ is allocated to container } i \\ 0 & \text{otherwise} \end{cases}$$

A cost C is associated to each pair container-task allocation. Its value depends on whether the container is overloaded after the task execution or not and also whether a task migration was necessary due to end user mobility. Energy consumption cost was not considered in this work. We present details about considered costs in the following:

3.1.2.1 Overload cost We denote by C_cap_i the computational capacity of container i at time t :

$$C_cap_i = \begin{pmatrix} C_cap_i^{CPU} \\ C_cap_i^{RAM} \\ C_cap_i^{Net} \end{pmatrix}$$

$C_ut_{j,i}$ is the average resource utilization of task j on container i :

$$C_ut_{j,i} = \begin{pmatrix} C_ut_{j,i}^{CPU} \\ C_ut_{j,i}^{RAM} \\ C_ut_{j,i}^{Net} \end{pmatrix}$$

For each new task j to be executed in container i , we express the utilization rate μ_i of container i corresponding to this system configuration as the ratio of average

resource utilization of task j on container i by the computational capacity (CPU, RAM, network) of the container:

$$\mu_i = \begin{pmatrix} \mu_i^{CPU} = \frac{\sum_{j=1}^T t_{(i,j)} C_ut_{j,i}^{CPU}}{C_cap_i^{CPU}} \\ \mu_i^{RAM} = \frac{\sum_{j=1}^T t_{(i,j)} C_ut_{j,i}^{RAM}}{C_cap_i^{RAM}} \\ \mu_i^{Net} = \frac{\sum_{j=1}^T t_{(i,j)} C_ut_{j,i}^{Net}}{C_cap_i^{Net}} \end{pmatrix} \quad (11)$$

If $\text{Max}(\mu_i^{CPU}, \mu_i^{RAM}, \mu_i^{Net}) > 1$, the container is considered in overloaded status. A penalty is associated when a task j is allocated on an overloaded container. We assume it to be positively proportional to the level of overloading. The overload cost ov_cost_i metric of a container i is defined as follows:

$$ov_cost_i = \begin{cases} (\mu_i - 1)^\lambda & \text{if } \text{Max}(\mu_i^{CPU}, \mu_i^{RAM}, \mu_i^{Net}) > 1 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

λ allows to accentuate ov_cost when the container is approaching its saturation state. Indeed, the closer the container is approaching its maximum capacity, the more the ov_cost will increase, and therefore, the algorithm will choose another less loaded container to execute the tasks and avoid overloading.

The overall overload cost for the Cloud-RRH system to execute all the offloading request tasks can be calculated using this expression:

$$ov_cost = \sum_{i=1}^N ov_cost_i, N : \text{set of containers} \quad (13)$$

3.1.2.2 Network delay cost The network delay is caused by processing, queuing, transmission, and propagation delays. It causes performance degradation to system users. We associate a per unit of time delay cost $d_{i,j}$ of task j allocation on container i . The overall network delay cost is:

$$nd_cost = \sum_i \sum_j t_{(i,j)} d_{i,j} \quad (14)$$

3.1.2.3 Migration cost A task can be migrated when the corresponding mobile user is moving from one cell to another one. If a user task j is migrated, from one

container to another one, a penalty r_j is associated in order to capture the service downtime incurred by this migration. The overall migration cost is defined as follows:

$$\text{mig.cost} = \sum_i \sum_j t_{(i,j)} r_j \quad (15)$$

Note that we only consider here a migration of the tasks in the same Cloud-RRH and that migration penalty only depends on the type of task.

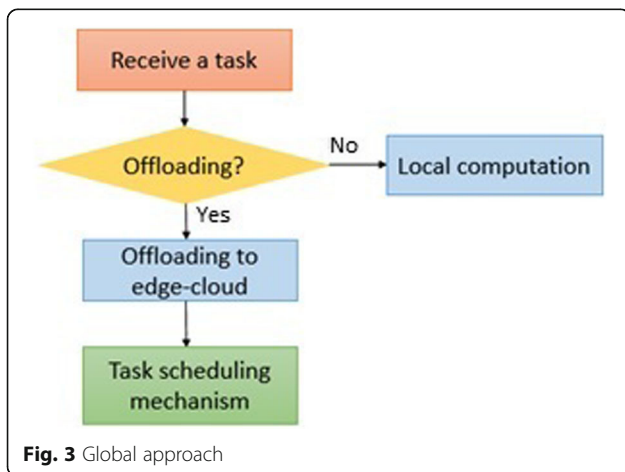
3.2 Joint offloading decision and task scheduling mechanism

In this section, we will present our strategy for joint offloading and tasks scheduling in 5G cloud radio access networks. The global idea is represented in Fig. 3. When a task is received, we will run an offloading decision algorithm considering a multitude of parameters about the mobile device capabilities, user mobility, and network status. If the decision is offloading to the edge cloud, a task scheduling mechanism is launched in order to improve resource utilization while reducing the execution cost.

Offloading mechanism is represented in Fig. 4. After receiving a task, the mobile terminal (MT) starts by generating an offloading request packet and sends it to the serving RRH. The offloading request packet has the following structure:

- *Offloading Req* (service ID, C_{MT} , C , E_{loc} , B), where C_{MT} is the capacity of the mobile terminal, C is the capacity required by the received application, E_{loc} is the energy that will be spent for local execution, and B is the bandwidth between the serving RRH and mobile device.

The offloading request is then sent to corresponding Cloud-RRH where the container's manager (CM) will decide about offloading or not using the offloading decision algorithm which will be detailed in the next session.



If the application will be offloaded in the Cloud-RRH, the CM sends a *Resource Allocation Request* packet to the serving container where it indicates the capacity required for processing. The Offloading Response will be routed up to the mobile device after execution.

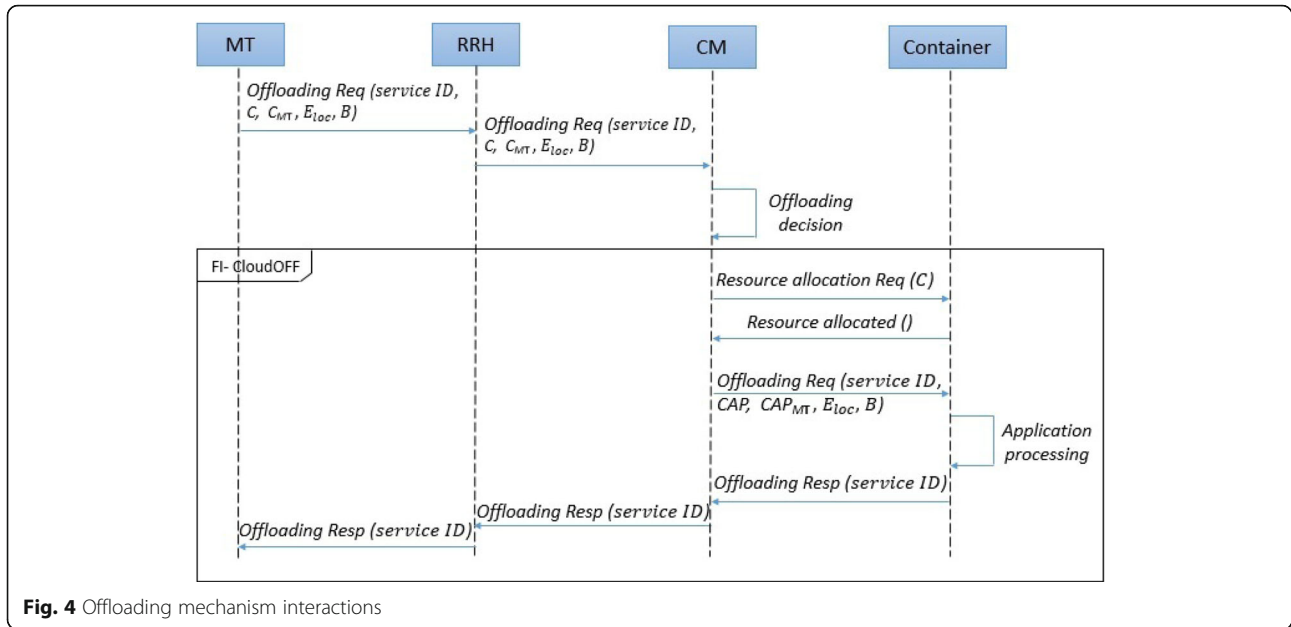
We propose a multi-parameter offloading decision algorithm which indicates where the application should be processed: locally on the mobile device, in the edge cloud (Cloud-RRH), or in central cloud (BBU pools). We aim to enhance the end user quality of experience (QoE) while improving the network and MT resource utilization. The decision algorithm workflow is represented in Fig. 5.

When a task is received, we will start by comparing the mobile device velocity to a velocity threshold (V_{th}): if the MT velocity is higher, the task will be executed locally. The main motivation to take into account the velocity in the offloading decision-making is to prevent mobile terminals moving with a high speed from offloading tasks in the cloud which may lead to quality of service degradation due to the degradation of the communication (e.g., handover) and the risk that the tasks are migrating too often between the access Cloud-RRHs. If not, we will test the latency due to local execution with the latency due to offloading using Eqs. (9) and (10). If the latency generated by offloading the task to Cloud-RRH is greater than the latency generated by local processing, the task should be executed locally at the mobile terminal. However, while mobile terminal computation resources are limited, if the computational capacity required is greater than the predefined percentage of the total locally available capacity, the task will be offloaded to the Cloud-RRH.

Otherwise, if the latency generated by offloading is lower, we have to compare the energy consumed by mobile terminal in case of offloading using Eq. (7) to the energy consumed in local execution case using Eq. (8). If E_{off} is higher than E_{loc} , we compare the latency generated by local computation to the maximum latency authorized by the application. If $L_{loc} < L_{max}$, the task is executed locally, if not, the task is offloaded to Cloud-RRH.

Finally, if $E_{off} < E_{loc}$, we will test the channel conditions. We can use the Shannon theorem to calculate the channel capacity. We consider that channel gain encompasses path loss, slow fading, and fast fading. Then, the channel coefficient is compared to the average channel coefficient calculated and updated over time. The channel is considered in a relatively "good" state if the current channel realization is above this average; thus, the task is offloaded. Otherwise, the task is executed locally. This will prevent the system from applying costly offloading when channel conditions are not favorable.

When a task is offloaded in the edge cloud, the container's manager will decide in which container the



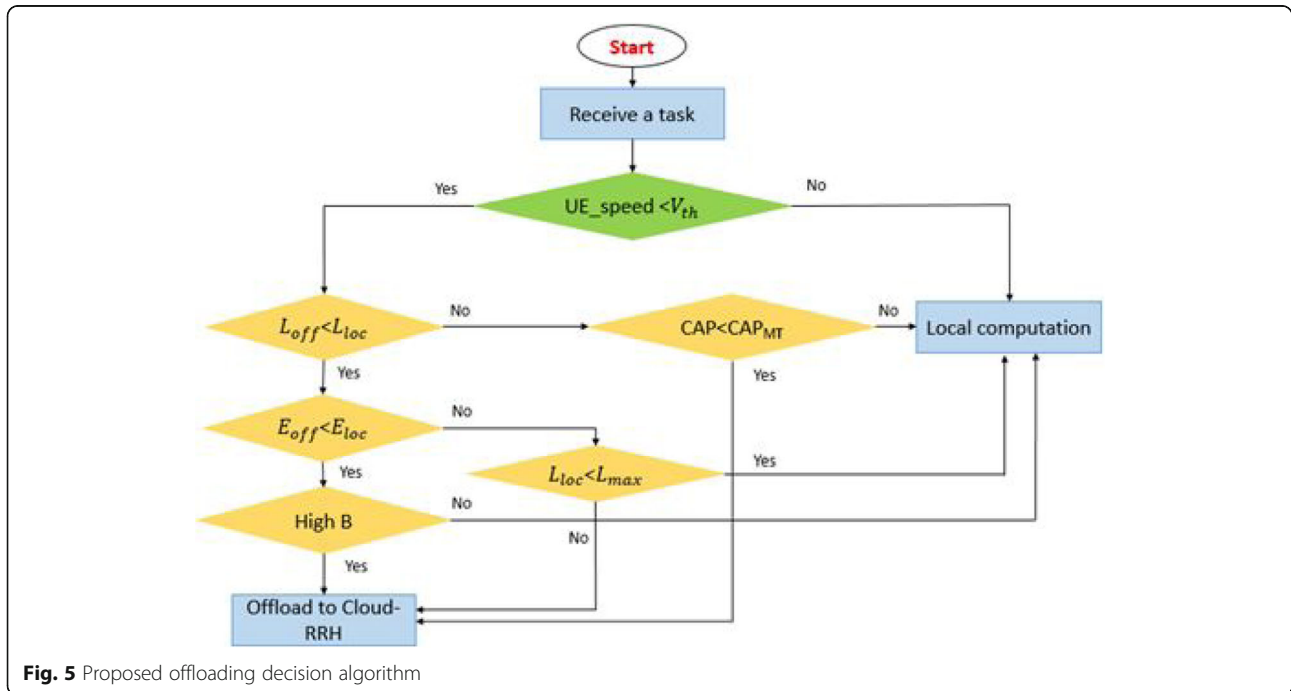
application will be processed. A container is characterized by a triplet of allocated resources (CPU, RAM, and network bandwidth). Each offloading request is considered as a set of tasks to instantiate in the Cloud-RRH. Each task has a delay constraint and is characterized by its resource requirements in terms of CPU, RAM, and network bandwidth.

It is necessary to well design the scheduler of tasks based on the available resources and tasks requirements in order to find the most suitable container for

application task offloading that minimizes the total cost while respecting load balancing between containers in the same Cloud-RRH. The total execution cost is expressed as follows:

$$C = \alpha \cdot ov_cost + \beta \cdot nd_cost + \delta \cdot mig_cost \quad (16)$$

α , β , and δ introduce the importance of weights associated to each cost to optimize. If the weights are equal, this means that there is no preference of one resource



against the others; otherwise, the resources that are assigned the highest weight will have the highest priority in the optimization process.

Therefore, the objective is to minimize the total cost of overloading, network delay, and migration of the entire system when executing all the submitted offloading requests.

Objective function

Minimize C

Subject to

$$\sum_j t_{(i,j)} T_{ex_j} \leq D \quad (17)$$

$$\begin{cases} \sum_j t_{(i,j)} C_{ut_{j,i}}^{CPU} \leq CPU_i \\ \sum_j t_{(i,j)} C_{ut_{j,i}}^{RAM} \leq RAM_i \\ \sum_j t_{(i,j)} C_{ut_{j,i}}^{Net} \leq Net_i \end{cases} \quad (18)$$

$$\left| \frac{\sum_i \mu_i}{N} - \mu_i \right| \leq \varepsilon \quad (19)$$

$$\sum_j t_{(i,j)} = 1 \quad (20)$$

The optimization is subject to constraints given by (17) through (20). Deadline constraint is expressed by Eq. (17). It guarantees that each offloading request is executed before the application's deadline. Equation (18) expresses resources constraint. It enforces that container resources are greater than all tasks' requirements including the number of CPU, amount of memory, and network bandwidth. The constraint (19) assures load balancing between containers in the same Cloud-RRH, and ε denotes for the maximum tolerance of load balancing. Finally, the constraint (20) guarantees that each task is scheduled on only one container.

We firstly consider that there is no preference between the different types of resources, i.e., $\alpha = \beta = \delta = 1$. We also consider that all tasks are executed in parallel and the deadline D constraint is therefore fixed for the worst case, that is, all tasks are executed in serial.

This is a MIP (mixed-integer problem) problem, and we solve it as a linear program since the objective function is linear to all variables.

4 Performances evaluation

In order to test our resource management scheme, we considered an urban environment simulation scenario. Our heterogeneous C-RAN infrastructure is composed of seven H-RRHs and four L-RRHs per cell. H-RRHs

have a coverage of 500 m and L-RRHs 30 m radius [23]. We have aligned values of ε_0 and t_0 with the measurements given in [24] for energy and frequency characteristics of local computing in commercial mobile handsets, as well as computation of data ratios in practical applications. As in an urban area, we have considered users with random velocity from 3 to 120 km/h.

In the Cloud-RRH, containers have a computing capacity from 25 to 100. The memory varies from 100 to 200 KB and the network bandwidth is set from 1 to 2 Kbps. The number of tasks T varies from 20 to 140, and they have heterogeneous requirements. To evaluate our cost-based task scheduling scheme, we compare its performances to the SAH-DB mechanism. System simulation parameters are listed in Table 1.

To evaluate our offloading decision algorithm performances, we compare it to the following algorithm:

- No offloading: all tasks are executed locally on the mobile handset
- Total offloading: all tasks are offloaded to the Cloud-RRH
- **SM-POD** [25]: task offloading is based on a series of successive classifications considering mobile terminal capacities, task characteristics, or communication channel state. As a first step, tasks are classified as offloadable or not regarding their

Table 1 Simulation parameters

| Parameters | Values |
|---|--|
| $k(tx,1)$ | 0.4 W |
| $k(tx,2)$ | 18 |
| $k(rx,1)$ | 0.4 W |
| $k(rx,2)$ | $2.86 \cdot 10^{-3}$ W/Mbps |
| ε_0 | $8.6 \cdot 10^{-8}$ J/bit |
| t_0 | 10^{-7} s/bit |
| t_1 | $t_0/2$ |
| Bandwidth (B) | 10 MHz |
| Maximum latency authorized by the application (L_{max}) | 4 s |
| Users mobility speed | $3 \text{ km/h} < V \leq 120 \text{ km/h}$ |
| V_{th} | 5 m/s |
| Number of containers | 25 |
| Containers' CPU variation | 1–10 |
| Containers' RAM variation | 128–512 MB |
| Containers' net variation | 100–200 Kbps |
| Number of tasks | 20–140 |
| Tasks' CPU variation | 1–4 |
| Tasks's RAM variation | 128–1024 KB |
| Tasks's Net variation | 1–20 Kbps |

characteristics. Then, offloadable and non-offloadable tasks are divided into urgent and not urgent tasks based on their latency requirements. The third step of the decision algorithm concerns only offloadable urgent tasks. It consists in checking resources. If there is a lack of resources in the mobile terminal, the task should be offloaded. Therefore, tasks are divided into offloadable urgent tasks that should be offloaded (SOffUrg) in priority and offloadable urgent tasks that could be either offloaded or executed locally on the mobile terminal (COffUrg) depending on the available resources in the terminal. In the fourth step, energy spent for local computation and energy consumed for offloading are compared in order to determine COffUrg tasks that have to be offloaded. Finally, channel state is checked in order to determine if offloadable non-urgent tasks have to be offloaded or deferred.

In order to evaluate the impact of users' speed on the quality of service, we have tested our system performances under varying velocity values. Figure 6 represents the application response time and MT energy consumption over mobile terminals speed (ranging from 5 to 30 km/h). We considered an application's data size of 100 Mb. We can see that the QoS decreases when users' velocity exceeds 15 km/h. Therefore, in order to prevent QoS degradation, the velocity threshold must be taken between 15 and 20 km/h; otherwise, offloading will generate a lot of overhead leading to longer response time and higher energy consumption.

Figure 7 illustrates the variation of the response time over the data size (ranging from 1 to 100 Mb). We can observe that the proposed offloading scheme can ameliorate the user experience by reducing the response time. For small data size, no offloading has the best performance because mobile terminal

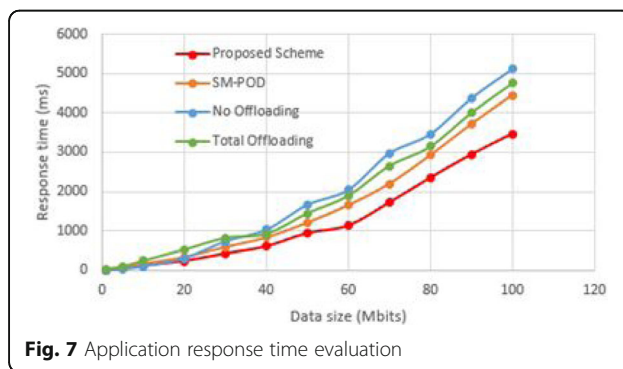


Fig. 7 Application response time evaluation

capacity is able to satisfy the application requirements. However, when the data size becomes larger, the difference between different schemes becomes larger. Thanks to cloud edge introduction and network flexibility, the proposed scheme has the lowest response time especially for big data size values. Thus, the proposed offloading decision algorithm can be useful for high resource demand applications.

Figure 8 shows the simulation results of the energy spent by the MT under the data size (ranging from 1 to 100 Mb). We can see that the proposed offloading decision algorithm can make the mobile handset consume less energy. The difference is more important when the data size is big. Therefore, the proposed algorithm is able to augment the mobile handset battery lifetime while executing complex program applications compared to local execution and total offloading.

When a task is offloaded to Cloud-RRH, we have evaluated the scheduling efficiency in terms of execution cost under a varying number of associated tasks. Figure 9 represents the execution cost by applying the proposed cost-based scheduling scheme and SAH-DB scheduling algorithm with 25 to 100 cloud containers respectively. The proposed

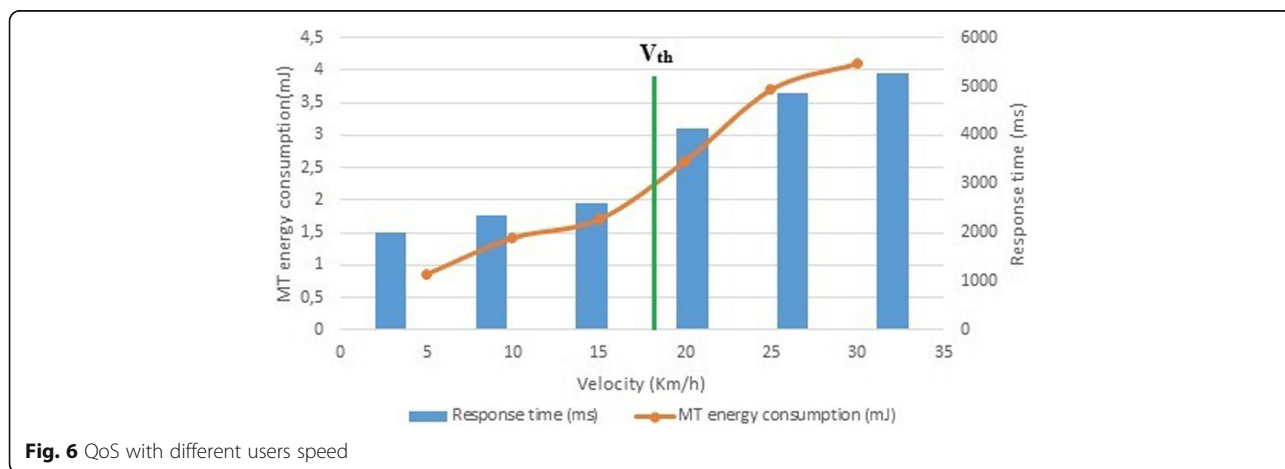


Fig. 6 QoS with different users speed

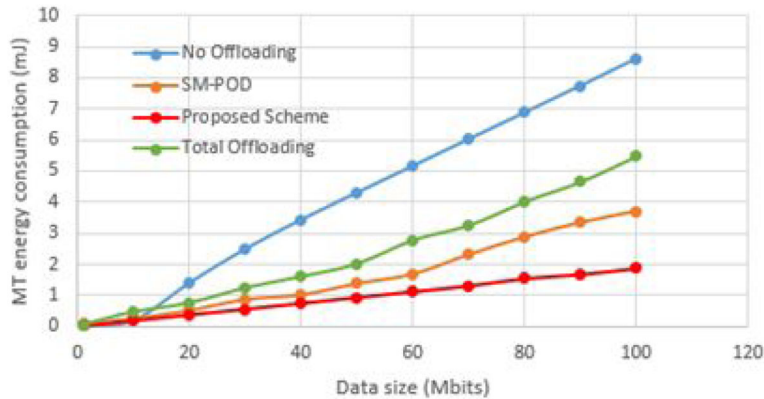


Fig. 8 Mobile terminal energy consumption evaluation

scheduling algorithm can reduce total execution cost compared with SAH-DB algorithm in the different number of associated tasks. Meanwhile, with the increase of the number of resources, the total cost of scheduling decreases. Moreover, the cost of scheduling increases with the number of associated tasks.

5 Conclusions

Data computation offloading enables intelligent mobile devices to run greedy resource applications. However, in order to fully exploit computation offloading benefits, we have to deal with two key challenges. The first one is about offloading decision, and the second one is about offloading request scheduling among available cloud resources [18]. The main contribution of this paper is the development of a whole offloading

strategy for H-CRAN composed of offloading decision and task scheduling in order to improve network performances and user QoE. Therefore, we jointly handle offloading decision and offloading request scheduling in Cloud-RRH. First, we proposed a dynamic multi-parameter offloading decision scheme in order to adapt offloading decision to the current network state and application characteristics. Then, scheduling mechanism was developed as linear programming optimization function that aims to reduce the total execution cost expressed as overload, network delay, and migration. Simulation results show that the proposed approach is able to improve QoS by reducing application response time and mobile device energy consumption while decreasing total execution cost in the Cloud-RRH.

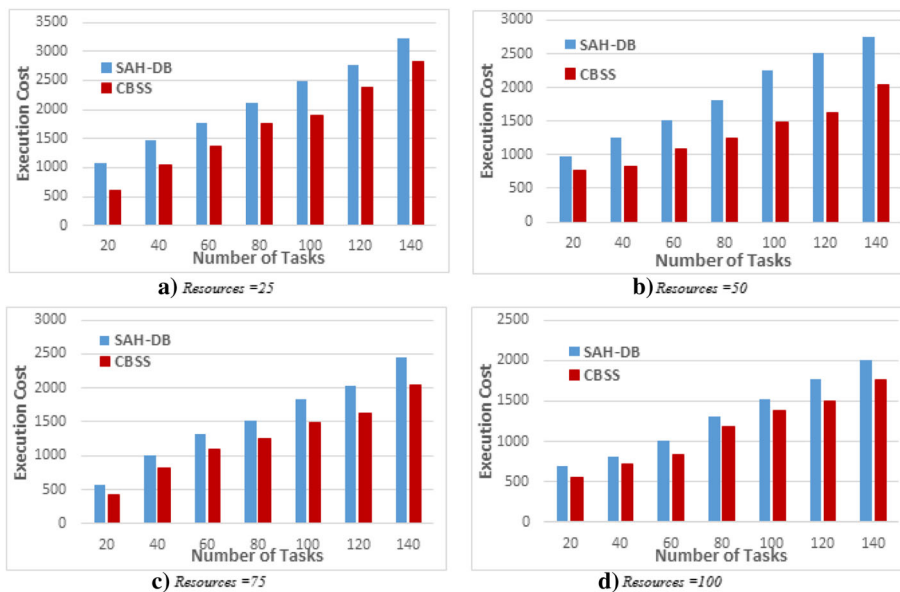


Fig. 9 The execution cost with different resources

Acknowledgements

The authors declare that no one contributes to this work except the authors.

Authors' contributions

All authors contributed to the work. All authors read and approved the final manuscript.

Funding

Only the authors financed this work.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹High School of Communication of Tunis (Sup'com), Carthage University, Ariana, Tunisia. ²IBISC-IBGBI Laboratory, University of Evry-Val-d'Essonne, Evry, France.

Received: 7 April 2017 Accepted: 5 November 2017

Published online: 21 November 2017

References

- E Cuervo, A Balasubramanian, D Cho, A Wolman, S Saroiu, R Chandra, P Bahl, in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, New York, NY, USA*. MAUI: making smartphones last longer with code offload (2010), pp. 49–62
- N Chalaemwongwan, W Kurutach, in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. Mobile cloud computing: a survey and propose solution framework (2016), pp. 1–4
- M Satyanarayanan, P Bahl, R Caceres, N Davies, The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **8**(4), 14–23 (2009)
- "Mobile-Edge Computing—Introductory Technical White Paper," Sep-2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf. Accessed 16 Mar 2017
- S Barbarossa, S Sardellitti, PD Lorenzo, in *2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. Joint allocation of computation and communication resources in multiuser mobile cloud computing (2013), pp. 26–30
- S Kosta, A Aucinas, P Hui, R Mortier, X Zhang, in *2012 Proceedings IEEE INFOCOM*. ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading (2012), pp. 945–953
- B-G Chun, S Ihm, P Maniatis, M Naik, A Patti, in *Proceedings of the Sixth Conference on Computer Systems, New York, NY, USA*. CloneCloud: elastic execution between mobile device and cloud (2011), pp. 301–314
- G Chen, BT Kang, M Kandemir, N Vijaykrishnan, MJ Irwin, R Chandramouli, Studying energy trade offs in offloading computation/compilation in Java-enabled mobile devices. *IEEE Trans. Parallel Distrib. Syst.* **15**(9), 795–809 (2004)
- K Liu, J Peng, X Zhang, Z Huang, in *2016 IEEE Global Communications Conference (GLOBECOM)*. A combinatorial optimization for energy-efficient mobile cloud offloading over cellular networks (2016), pp. 1–6
- T-Y Lin, T-A Lin, C-H Hsu, C-T King, in *2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Context-aware decision engine for mobile cloud offloading (2013), pp. 111–116
- YS Chen, CS Hsu, TY Juang, HH Lin, in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. An energy-aware data offloading scheme in cloud radio access networks (2015), pp. 1984–1989
- J Cheng, Y Shi, B Bai, W Chen, in *2016 IEEE International Conference on Communications (ICC)*. Computation offloading in cloud-RAN based mobile cloud computing system (2016), pp. 1–6
- T Zhao, S Zhou, X Guo, Y Zhao, Z Niu, in *2015 IEEE Globecom Workshops (GC Wkshps)*. A cooperative scheduling scheme of local cloud and Internet cloud for delay-aware mobile cloud computing (2015), pp. 1–6
- Z Huang, B Balasubramanian, M Wang, T Lan, M Chiang, DHK Tsang, in *2015 IEEE Conference on Computer Communications (INFOCOM)*. Need for speed: CORA scheduler for optimizing completion-times in the cloud (2015), pp. 891–899
- M NoroozOliaee, B Hamdaoui, M Guizani, MB Ghorbel, in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Online multi-resource scheduling for minimum task completion time in cloud servers (2014), pp. 375–379
- Himani, HS Sidhu, in *2015 Second International Conference on Advances in Computing and Communication Engineering*. Cost-deadline based task scheduling in cloud computing (2015), pp. 273–279
- M Yingchi, X Ziyang, P Ping, W Longbao, in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. Delay-aware associate tasks scheduling in the cloud computing (2015), pp. 104–109
- O Chabbouh, SB Rejeb, Z Choukair, N Agoulmine, in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. Offloading decision algorithm for 5G/HetNets cloud RAN (2016), pp. 1–5
- O Chabbouh, SB Rejeb, N Agoulmine, Z Choukair, in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. Cloud RAN architecture model based upon flexible RAN functionalities split for 5G networks (2017), pp. 184–188
- S Barbarossa, S Sardellitti, PD Lorenzo, in *2013 Future Network Mobile Summit*. Computation offloading for mobile cloud computing based on wide cross-layer optimization (2013), pp. 1–10
- D Huang, P Wang, D Niyato, A dynamic offloading algorithm for mobile computing. *IEEE Trans. Wirel. Commun.* **11**(6), 1991–1995 (2012)
- O Muñoz, A Pascual-Iserte, J Vidal, in *2013 Future Network Mobile Summit*. Joint allocation of radio and computational resources in wireless application offloading (2013), pp. 1–10
- J Oueis, EC Strinati, S Barbarossa, in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. Multi-parameter decision algorithm for mobile computation offloading (2014), pp. 3005–3010
- AR Jensen, M Lauridsen, P Mogensen, TB Sørensen, P Jensen, in *2012 IEEE Vehicular Technology Conference (VTC Fall)*. LTE UE power consumption model: for system level energy and performance optimization (2012), pp. 1–5
- AP Miettinen, JK Nurminen, in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, Berkeley, CA, USA*. Energy efficiency of mobile clients in cloud computing (2010), pp. 4–4

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com