Green Cloudlet Network: A Sustainable Platform for Mobile Cloud Computing

Xiang Sun, *Student Member, IEEE,* and Nirwan Ansari, *Fellow, IEEE*

In the Green Cloudlet Network (GCN) architecture, each User Equipment (UE) is associated with an Avatar (a private virtual machine for executing its UE's offloaded tasks) in a cloudlet located at the network edge. In order to reduce the operational expenditure for maintaining the distributed cloudlets, each cloudlet is powered by green energy and uses on-grid power as a backup. Owing to the spatial dynamics of energy demands and green energy generations, the energy gap (i.e., energy demand minus green energy generation) among different cloudlets in the network is unbalanced, i.e., some cloudlets' energy demands can be fully provisioned by their green energy generations but others need to utilize on-grid power to meet their energy demands. The unbalanced energy gap increases the on-grid power consumption of the cloudlets. In this paper, we propose the Green-energy aware Avatar Placement (GAP) strategy to minimize the total on-grid power consumption of the cloudlets by migrating Avatars among the cloudlets according to the cloudlets' residual green energy, while guaranteeing the service level agreement (the End-to-End (E2E) delay requirement between a UE and its Avatar). Simulation results show that GAP can save 57.1% and 57.6% of on-grid power consumption as compared to the two other Avatar placement strategies, i.e., Static Avatar Placement and Follow me AvataR, respectively.

***Index Terms*—Mobile cloud computing, cloudlet, energy optimization, Avatar placement, migration.**

I. Introduction

The emergence of Mobile Cloud Computing (MCC) [1], [2] is enabling User Equipments (UEs) to offload their computation-intensive tasks (e.g., augmented reality, natural language translation, face and object recognition, dynamic activity interpretation, and body language interpretation) to a data center, which provisions flexible resource allocation and efficient parallel computing [3], [4]. The data center can help UEs execute these tasks to reduce the task execution time and the energy consumption of UEs. However, the existing MCC architecture, as shown in Fig. 1, suffers from the long End-to-End (E2E) delay between a UE and a remote data center because the communications between a UE and the remote data center usually traverses the Internet, which incurs unbearable and uncontrollable E2E delay (especially during peak hours). Note that the E2E delay is a critical Quality of Service (QoS) parameter in provisioning MCC applications. It is reported that augmented reality applications require an E2E delay of less than 16 ms [5] and virtual desktop applications require an E2E delay of less than 60 ms [6].

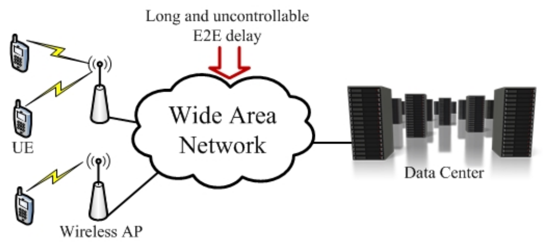


Fig. 1. The existing MCC architecture.

X. Sun and N. Ansari are with the Advanced Networking Lab., Helen and John C. Hartmann Department of Electrical & Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA. E-mail:{xs47, nirwan.ansari}@njit.edu.
This work was supported in part by NSF under grant no. CNS-1320468.

In order to maintain low E2E delay between UEs and computing resources, a green cloudlet network architecture [7] was proposed, in which computing resources are placed close to UEs. Specifically, as shown in Fig. 2, each Base Station (BS) in the mobile network is connected to a cloudlet, which is considered as a self-maintained tiny data center comprising a number of interconnected Physical Machines (PMs) [8], [9]. Since BSs have already been deployed in the network and provide seamless wireless connections for UEs, UEs can actually offload their computation-intensive tasks to nearby cloudlets via one wireless hop [10]. This can tremendously reduce the E2E delay between UEs and computing resources. In order to preserve privacy when UEs offload their tasks to cloudlets, each UE is associated with a dedicated Avatar to execute its offloaded tasks [11]. An Avatar is considered as a private high performance Virtual Machine (VM), which comprises two parts, i.e., *baseVM* and *overlayVM*. *baseVM* is a minimally configured guest Operating System (OS) running on top of hardware implementations in a PM; normally, the guest OS of an Avatar is the same as the one of its UE such that the Avatar can run unmodified application components in its UE [12]. *overlayVM*, whose size is much smaller than *baseVM*, contains the private data of its UE, such as the MCC applications and the states of these MCC applications. The detailed configuration of an Avatar can be found in [13].

Note that Avatars can not only execute the offloaded tasks from their UEs but also analyze the data streams sensed by their UEs [14]. A typical example of utilizing GCN to analyze the big data is the terrorist localization application [15], which is to identify and trace terrorists by analyzing the photos and videos captured by UEs. Specifically, the terrorist localization application (which is to identify and trace terrorists) would send the terrorists' photos to each Avatar, which runs the face matching algorithm locally to compare the recent photos and videos (which are captured and uploaded from UEs) with the terrorists' photos. If matched, the information (i.e., the locations and timestamps) of the photos/videos would be

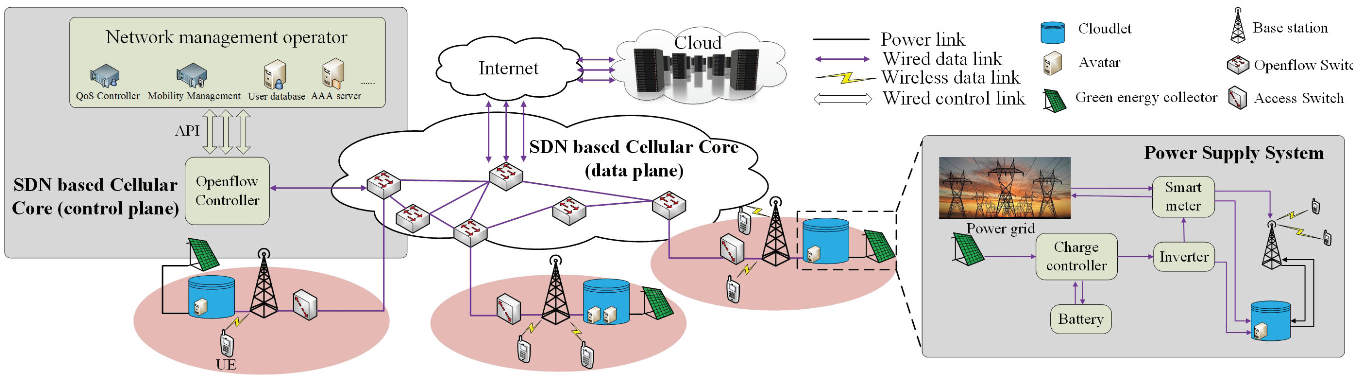


Fig. 2. The GCN architecture.

uploaded to the terrorist localization application for further processing. As compared to the traditional method (where all the captured photos and videos would be uploaded to a remote data center for further processing), GCN based distributed big data processing can substantially reduce the traffic of the network and the response time for identifying terrorists. Therefore, an Avatar plays two roles in GCN, i.e., the big data analyzer and the MCC application outsourcer of its UE.

On the top of cloudlets and BSs, Software Defined Network (SDN) based cellular core network [16], [17] has been introduced to replace the traditional cellular core network to establish efficient and flexible communications paths between Avatars in different cloudlets as well as between UEs in different BSs. In the SDN based cellular core, a number of OpenFlow switches are interconnected together to provision data plane functionalities (e.g., packet routing). All the control functions are extracted from OpenFlow switches and centralized in the OpenFlow controller, which controls all the OpenFlow switches based on the OpenFlow protocol [18]. The OpenFlow controller manages the forwarding plane of BSs and OpenFlow switches, monitors the traffic at the data plane, and establishes user sessions. Also, it provides application programming interfaces (APIs) to network management operators so that different network functionalities, such as mobility management, user authentication, authorization and accounting, network visualization, and QoS control, can be added, removed, and modifed flexibly. Moreover, every UE and its Avatar in a cloudlet can communicate with public data centers via the Internet to provision scalability, i.e., if cloudlets are not available for UEs because of the capacity limitation, Avatars can be migrated to the remote data centers to continue serving their UEs.

GCN facilitates the application workloads offloading process as well as big data networking, but maintaining a number of distributed cloudlets incurs a huge operational expenditure (OPEX) to the cloudlet provider by paying an expensive energy bill to the on-grid power suppliers. In order to reduce on-grid power[1], "greening" is introduced in GCN, i.e., each cloudlet is powered by green energy, which is considered as a "free" energy supply for the cloudlet provider, and uses on-grid power as a backup. The power supply system of each cloudlet is shown in Fig. 2, in which the green energy collector absorbs energy from renewal resources and converts it into electrical power, the charge controller regulates the electrical power from the green energy collector, and the electrical power is converted between AC and DC by the inverters. The smart meter records the electric energy from the power grid consumed by the cloudlet and BS.

[1]Note that we assume GCN is a time-slotted system, and so "power" and "energy" are interchangeable in the rest of the paper.

A. Avatar Migration in GCN

Avatars can be statically deployed in their initiated cloudlets even if their UEs roam far away from the BSs, whose connected cloudlets contain these Avatars. Static Avatar Placement (SAP) may result in long E2E delay between UEs and their Avatars. In order to minimize the E2E delay, Follow me AvataR (FAR), which is inspired by the idea in [19], is proposed to migrate a UE's Avatar to the cloudlet (whose connected BS is serving the UE) when the UE roams away. As shown in Fig. 3, when a UE moves from BS-A's coverage area into BS-B's coverage area, its Avatar is migrated from Cloudlet A into Cloudlet B such that the E2E delay between the UE and its Avatar can still be maintained at a low level.

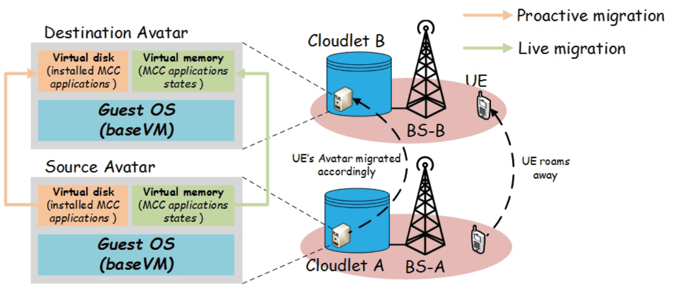

Proactive migration
Live migration
Destination Avatar
Virtual disk (installed MCC applications)
Virtual memory (MCC applications states)
Guest OS (baseVM)
Source Avatar
Virtual disk (installed MCC applications)
Virtual memory (MCC applications states)
Guest OS (baseVM)
Cloudlet B
UE
BS-B
UE's Avatar migrated accordingly
UE roams away
Cloudlet A
BS-A

Fig. 3. The illustration of migrating Avatars among cloudlets to reduce the energy consumption from the power grid.

The Avatar migration process comprises two steps. First, the guest OS (i.e., *baseVM*) is launched in the destination Avatar. This can be achieved by waking up an asleep VM installed with the corresponding guest OS. Second, the *overlayVM* from the source Avatar is migrated to the destination Avatar. As mentioned before, the *overlayVM* comprises the MCC applications (which are installed in the source Avatar) and the states of these MCC applications. The MCC applications are stored in the virtual disk of the source Avatar and can be proactively migrated to the destination Avatar. Specifically, a UE is predicted to be in BS-B's coverage area in the next time slot, and thus a destination Avatar in Cloudlet B would be created by launching the guest OS and transferring the virtual disk of the source Avatar to the destination Avatar in the current time slot. The states of the MCC applications are stored in the virtual memory of the source Avatar. The virtual memory is migrated when the Avatar migration is triggered in the next time slot. Once the Avatar migration is completed (i.e., the virtual memory and virtual disk of the source Avatar have been transferred to the destination Avatar), the destination Avatar, which incurs the low E2E delay to its UE, would start to serve its UE by executing the offloaded tasks.

B. *Unbalanced Energy Gap Among Cloudlets in GCN*

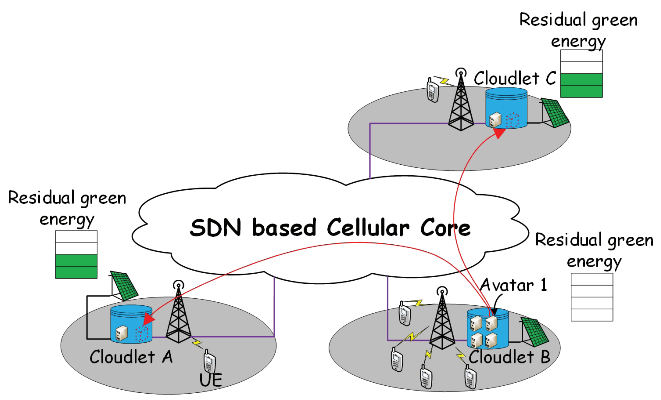


Fig. 4. The illustration of migrating Avatars among cloudlets to reduce the energy consumption from the power grid.

Owing to the spatial dynamics of the distribution of UEs among different BSs' coverage areas and the dynamics of application workloads among Avatars, different cloudlets may demand different amounts of energy for running the application workloads of the hosting Avatars. Meanwhile, green energy generation also exhibits spatial dynamics among different cloudlets [20], [21]. Therefore, some cloudlets, which have less energy demand and more green energy generated, would have excess of green energy. Conversely, some cloudlets, which have more energy demands and less green energy generated, would pull energy from the power grid. Such unbalanced energy gap (i.e., energy demand minus green energy generation) among different cloudlets increases the total on-grid power consumption [22]. For instance, as shown in Fig. 4, suppose there are three cloudlets, i.e., Cloudlet A, Cloudlet B and Cloudlet C, and different cloudlets host different numbers of Avatars in serving their local UEs. Assume that each Avatar consumes one unit of energy and each cloudlet generates three units of green energy in a time slot. Obviously, Cloudlet B does not have enough green energy to host four Avatars and needs to pull one unit of energy from the power grid. However, if one of the Avatars in Cloudlet B (e.g., Avatar 1) can be migrated to one of the other two cloudlets, no extra energy from the power grid needs to be drawn. Therefore, fully utilizing green energy can reduce the on-grid power consumption in GCN, thus decreasing the OPEX of the cloudlet provider.

In this paper, we are focusing on designing an efficient Avatar placement strategy in the context of GCN by migrating Avatars from the cloudlets with positive energy gap (i.e., with energy demand higher than green energy generation) into the cloudlets with negative energy gap (i.e., with energy demand lower than green energy generation). Thus, the total on-grid power consumption of GCN is minimized and the Service Level Agreement (SLA), which is defined as the E2E delay requirement between the UE and its Avatar, is guaranteed.

The rest of the paper is organized as follows. In Section II, we briefly review the related works. In Section III, we set up a power consumption model of a cloudlet in GCN and an E2E delay model between a UE and its Avatar in the cloudlet, upon which we formulate the Avatar placement problem to minimize the total on-grid power consumption, and prove its NP hardness. In Section IV, we propose the Green-energy aware Avatar Placement (GAP) heuristic algorithm to solve the problem. In Section V, we demonstrate the performance and scalability of GAP via simulation results. The conclusion is presented in Section VI.

II. Related works

Offloading computing intensive tasks from resource constrained UEs to other computing facilities is intriguing to reduce the energy consumption of UEs and task execution time. Li and Wang [23] proposed that a UE (i.e., an initiator) can offload its tasks to nearby UEs (i.e., UEs within one wireless hop coverage to the initiator). However, owing to the mobility of UEs and randomness of inter-contact time between the initiator and other UEs, the initiator can only offload delay tolerant tasks to nearby UEs. To overcome the drawbacks of randomness of accessing computing facilities, the Mobile Edge Computing (MEC) concept has been proposed to enable a UE to offload its computing intensive tasks to facilities placed at the mobile edge. Currently, many MEC frameworks have been designed to achieve task offloading process to benefit UEs. Satyanarayanan *et al.* [24] first proposed to apply a cloudlet to execute tasks offloaded from local UEs. They designed a system named Kimberley [24], whose architecture is applied to design Avatars, to facilitate task offloading. The MAUI project [25] provides method level code offloading based on the .NET framework. Different from Kimberley, MAUI provides a method to enable each UE in determining whether to offload the source codes of an application (based on some context information, i.e., the computing resource demands, execution time, network condition, and state transfer requirements) such that the energy consumption of the UE is minimized. CloneCloud [26] and ThinkAir [27] are designed for Java applications written for Android based UEs in offloading their tasks to cloudlets. ThinkAir focuses on how to efficiently and flexibly request computing resources in cloudlets, while CloneCloud takes the advantage of high compatibility, i.e., source codes of mobile applications can be executed in a VM (in a nearby cloudlet) without any modification. Instead of designing MEC frameworks to maximize the benefits from UEs, Hoang *et al.* [28] designed an admission control scheme of a cloudlet to determine which offloaded tasks should be executed in the cloudlet. The admission control scheme is to maximize the revenue for cloudlet service providers.

In order to solve the UE mobility problem (i.e., the E2E delay between a UE and its Avatar becomes unbearable when the UE roams far away from its original place), migrating Avatars among cloudlets based on their UEs' locations has been proposed to reduce the E2E delay. Ha *et al.* [6] demonstrated the feasibility of migrating an Avatar between two cloudlets over the local area network. The results show that the Avatar migration can be completed within one minute over a 25 $Mbps$ wired link. Sun and Ansari [16] proposed that Avatars can be migrated among the cloudlets over SDN based cellular core. They proposed an Avatar placement strategy to optimize the tradeoff between the migration gain and the migration cost. In order to reduce the Avatar migration cost in terms of the extra traffic volume generated during the migration, Sun and Ansari [29] proposed to place a number of replicas of an Avatar's *overlayVM* in the cloudlets, which are commonly visited by UEs. This can significantly reduce the migration time as well as the migration traffic. Our previous work [7] introduces the concept of the GCN architecture by powering each cloudlet with green energy to reduce the OPEX of cloudlet network providers. However, it does not provide the solution on how to fully utilize green energy among cloudlets.

As compared to the previous efforts, this paper has made several contributions.

1) We demonstrate the unbalanced energy gap among different cloudlets in GCN.
2) We formulate the Avatar placement problem, which is to minimize the total on-grid power consumption of GCN while guaranteeing the SLA in terms of the E2E delay bound for each UE and its Avatar. We prove the Avatar placement problem to be NP-hard.
3) We design a novel heuristic algorithm (i.e., GAP) to solve the problem efficiently.
4) As the results generated by GAP are comparable to those generated by CPLEX[2] (which is considered to be the optimal solution of the Avatar placement problem) in a small-scale network deployment, we demonstrate that GAP is a good heuristic algorithm to solve the problem.
5) We show the total amount of on-grid power consumption incurred by GAP is less than that incurred by other two Avatar placements, i.e., SAP and FAR. The results prove that GAP can significantly reduce the on-grid power consumption while guaranteeing the SLA.

III. System model

We assume that PMs in each cloudlet of GCN are homogeneous, i.e., the configuration of each PM is the same. Meanwhile, the configuration of each UE's Avatar is also homogeneous, but the application workloads vary among Avatars. Therefore, each PM can host a fixed number of Avatars, but the application workloads in different PMs vary.

A. Cloudlet power consumption

Let $\mathcal{I}$ and $\mathcal{K}$ be the sets of UEs and cloudlets, respectively. Denote x_{ik} as a binary variable to indicate whether UE i's Avatar ($i \in \mathcal{I}$) is located in cloudlet k (i.e., $x_{ik} = 1$, where $k \in \mathcal{K}$) or not (i.e., $x_{ik} = 0$). Meanwhile, we assume the power consumption of each PM is approximately linear with respect to the PM's CPU utilization [30], [31], i.e.,

$$p_m = p^{idle} + \alpha \mu_m, \tag{1}$$

where p_m is the power consumption of PM m, p^{idle} is the power consumption of the PM in the idle mode (i.e., the CPU utilization of the PM is zero), μ_m is the CPU utilization of PM m, and α is the power coefficient that maps the CPU utilization into power consumption. The power consumption of the cloudlet is equal to the sum of the awake PMs' power consumption. Since each PM can host a fixed number of Avatars ϵ, the number of the awake PMs in cloudlet k is:

$$M_k = \left\lceil \frac{\sum\limits_{i \in \mathcal{I}} x_{ik}}{\epsilon} \right\rceil, \tag{2}$$

where $\lceil \bullet \rceil$ is the ceiling function.

The power consumption of cloudlet k, denoted as p_k, is:

$$p_k = \sum_{m=1}^{M_k} p_m = M_k p^{idle} + \alpha \sum_{m=1}^{M_k} \mu_m. \tag{3}$$

Note that the total CPU utilization of the awake PMs in cloudlet k (i.e., $\sum\limits_{m=1}^{M_k} \mu_m$) is equal to the total CPU utilization of the Avatars located in cloudlet k, i.e.,

$$\sum_{m=1}^{M_k} \mu_m = \sum_{i \in \mathcal{I}} \mu_i x_{ik}, \tag{4}$$

where μ_i is the CPU utilization of UE i's Avatar. By approximating Eq. 2 into $M_k \approx \frac{\sum_{i \in \mathcal{I}} x_{ik}}{\epsilon}$, the power consumption of cloudlet k, i.e., Eq. 3, becomes:

$$p_k = \sum_{i \in \mathcal{I}} \left(\frac{p^{idle}}{\epsilon} + \alpha \mu_i \right) x_{ik}. \tag{5}$$

Based on Eq. 5, we define the power consumption of UE i's Avatar, denoted as ρ_i, as:

$$\rho_i = \frac{p^{idle}}{\epsilon} + \alpha \mu_i. \tag{6}$$

B. E2E delay model

Maintaining the low E2E delay between a UE and its Avatar is critical for MCC applications and real-time big data networking. The proposed GCN architecture has the potential to provide the low E2E delay between UEs and their Avatars. However, UEs are roaming among BSs over time, and so the E2E delay may worsen if their Avatars remain in their

[2]CPLEX is a common commercial solver to generate near optimal solutions of most optimization problems at the cost of long execution time and huge CPU as well as memory resource requirements. The detailed description of CPLEX can be found in https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

original cloudlets. Also, as mentioned in Sec. I, the E2E delay may become worse when Avatars migrate to the cloudlets with more residual green energy. Therefore, it is necessary to migrate the UE's Avatar to a suitable cloudlet if the E2E delay between the UE and its Avatar is larger than the predefined SLA (i.e., the E2E delay bound). Note that the E2E delay between a UE and its Avatar comprises three parts: first, the E2E delay between a UE and its serving BS; second, the E2E delay between the BS and the cloudlet which contains the UE's Avatar; third, the E2E delay within the cloudlet. Normally, the E2E delay between the BS and the cloudlet is the main factor in degrading the overall E2E delay. Thus, we refer to the E2E delay between a UE and its Avatar as the E2E delay between the BS (which is serving the UE) and the cloudlet (which contains the UE's Avatar) in the rest of the paper.

Let $\mathcal{J}$ be the set of BSs. Denote y_{ij} as a binary indicator to indicate whether UE i is in BS j's coverage area (i.e., $y_{ij}=1$) or not (i.e., $y_{ij}=0$). Note that it has been demonstrated that about 10% to 30% of all human movement can be accounted for by their social relationship, while 50% to 70% is attributed to periodic behaviors [32]; thus, we believe that the dynamics of future human movement can be reliably predicted based on the mathematical models [32]–[34]. Also, denote t_{jk} as the average E2E delay between BS j and cloudlet k. Note that the value of t_{jk} can be measured and recorded by the SDN controller periodically [35], [36], and if $j=k$, we say that cloudlet k is BS j's attached cloudlet. Thus, the E2E delay between UE i and its Avatar, denoted as τ_i, is:

$$\tau_i = \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} y_{ij}t_{jk}x_{ik}. \tag{7}$$

C. Problem formulation

As mentioned earlier, energy demands and green energy generations among different cloudlets exhibit spatial dynamics, and so the energy demands of some cloudlets can be met by green energy, but some cannot and need to consume on-grid power. Such unbalanced energy gap among the cloudlets results in more on-grid power consumption. Therefore, we propose to schedule the placement of Avatars in GCN in each time slot to balance the energy gap among the cloudlets, while guaranteeing the SLA for each UE. Denote G_k as the green energy generation of cloudlet k and denote n_k as the total number of PMs in cloudlet k; meanwhile, let φ be the SLA in terms of the E2E delay bound. Then, we formulate the problem as follows:

$$\boldsymbol{P0}: \arg\min_{x_{ik}} \sum_{k\in\mathcal{K}} \max\left\{\sum_{i\in\mathcal{I}}\left(\frac{p^{idle}}{\epsilon}+\alpha\mu_i\right)x_{ik}-G_k, 0\right\} \tag{8}$$

$$s.t.\quad \forall i\in\mathcal{I},\quad \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} y_{ij}t_{jk}x_{ik}\le\varphi, \tag{9}$$

$$\forall k\in\mathcal{K},\quad \sum_{i\in\mathcal{I}} x_{ik}\le\epsilon n_k, \tag{10}$$

$$\forall i\in\mathcal{I},\quad \sum_{k\in\mathcal{K}} x_{ik}=1, \tag{11}$$

$$\forall i\in\mathcal{I}\ \forall k\in\mathcal{K},\quad x_{ik}\in\{0,1\}, \tag{12}$$

where $\sum_{i\in\mathcal{I}}\left(\frac{p^{idle}}{\epsilon}+\alpha\mu_i\right)x_{ik}$ in Eq. 8 is the power consumption of cloudlet k, $\max\left\{\sum_{i\in\mathcal{I}}\left(\frac{p^{idle}}{\epsilon}+\alpha\mu_i\right)x_{ik}-G_k, 0\right\}$ is the on-grid power consumption of cloudlet k, and thus the objective is to minimize the total on-grid power consumption of all the cloudlets in GCN. Constraint (9) imposes the E2E delay between each UE and its Avatar not to be larger than the predefined SLA; Constraint (10) imposes the total number of Avatars assigned to the cloudlet not to exceed the cloudlet's capacity; Constraint (11) imposes each Avatar to be placed in only one cloudlet.

The solution to Problem $\boldsymbol{P0}$ is to determine the placement of each Avatar in the next time slot. If an Avatar will be placed in Cloudlet A in the next time slot but is placed in Cloud B in the current time slot, we say the Avatar will be migrated from Cloudlet A into Cloudlet B. The whole schedule of migrating the Avatar from Cloudlet A into Cloudlet B is shown in Fig. 5. Specifically, 1) if a UE's Avatar is determined to migrate from Cloudlet A into Cloudlet B, a new VM would be resumed in Cloudlet B and is considered as the destination Avatar (which will serve its UE in the next time slot) of the UE; 2) after resuming the destination Avatar, the source Avatar (which is currently serving the UE in Cloudlet A) starts to migrate its virtual disk (i.e., the installed MCC applications) to the destination Avatar in the current time slot. Migrating the virtual disk can be optimally scheduled, i.e., the virtual disk can be scheduled and migrated as long as the source Avatar currently has enough available bandwidth. Virtual disk migration needs to be completed during the current time slot; 3) once the next time slot starts, the source Avatar begins to migrate its virtual memory (i.e., the states of the installed MCC applications) to the destination Avatar; 4) after virtual memory migration finishes, the destination Avatar starts to serve the UE and the source Avatar may go to sleep.

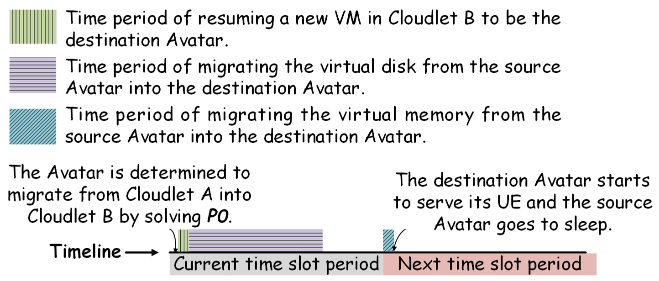

Time period of resuming a new VM in Cloudlet B to be the destination Avatar.
Time period of migrating the virtual disk from the source Avatar into the destination Avatar.
Time period of migrating the virtual memory from the source Avatar into the destination Avatar.
The Avatar is determined to migrate from Cloudlet A into Cloudlet B by solving P0.
The destination Avatar starts to serve its UE and the source Avatar goes to sleep.
Timeline
Current time slot period
Next time slot period

Fig. 5. Avatar migration schedule.

Note that we do not consider the energy consumption for conducting Avatar migration because the total energy consumption of each cloudlet depends on the energy consumption of its hosting PMs, whose energy consumption is mainly determined by their CPU utilization [30], [31], [41]–[43]. Yet, the Avatar migration process is considered as a network intensive application running on both source and destination Avatars [16], [44], i.e., the Avatar migration process does not significantly affect the CPU utilization of both source and destination Avatars, and thus the energy consumption for conducting Avatar migration is assumed to be negligible.

Theorem 1. *The problem of minimizing the on-grid power consumption of GCN (i.e., $\boldsymbol{P0}$) is NP-hard.*

Proof: Suppose there are 2 cloudlets in GCN (i.e., $|\boldsymbol{\mathcal{K}}| = 2$) and the capacity of each cloudlet is large enough to host all the UEs' Avatars in the network. Meanwhile, every Avatar can be placed in any of the two cloudlets without violating the SLA. Moreover, assume the green energy generation of each cloudlet is the same, and it is equal to half of the total energy demands of the network, i.e., $G_1 = G_2 = \frac{1}{2}\sum\limits_{i\in\boldsymbol{\mathcal{I}}}\left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i\right)$. Thus, $\boldsymbol{P0}$ can be transformed into:

$$\boldsymbol{P1}: \arg\min_{x_{ik}} \sum_{k=1}^{2} \max\left\{\sum_{i\in\boldsymbol{\mathcal{I}}}\left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i\right)x_{ik} - G_k, 0\right\}$$
$$s.t. \quad Constraints \ (11), (12).$$

Obviously, the optimal solution of $\boldsymbol{P1}$ is to place the Avatar into the two cloudlets so that the total energy demands of the two cloudlets are the same, i.e., $\sum\limits_{i\in\boldsymbol{\mathcal{I}}}\left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i\right)x_{i1} = \sum\limits_{i\in\boldsymbol{\mathcal{I}}}\left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i\right)x_{i2} = G_1 = G_2$. Thus, $\boldsymbol{P1}$ is equivalent to the partition problem, which is a well-known NP-hard problem. Therefore, we conclude that the partition problem is reducible to $\boldsymbol{P0}$, and so $\boldsymbol{P0}$ is NP-hard. ■

IV. Green-energy aware Avatar Placement (GAP)

In this section, we propose a heuristic algorithm, namely, Green-energy aware Avatar Placement (GAP), to find the suboptimal solution of $\boldsymbol{P0}$. Generally, GAP comprises two steps. In the first step, GAP tries to find the feasible Avatar placement, denoted as $\boldsymbol{\mathcal{X}} = \{x_{ik} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}}\}$, so that Constraints (9)–(12) are satisfied. In the second step, based on the Avatar placement generated in the first step, GAP tries to shift the energy demands (i.e., migrating Avatars) among cloudlets so that the on-grid power consumption is minimized while the SLA and each cloudlet's capacity requirement are satisfied.

A. Feasible Avatar placement

As mentioned previously, GAP tries to find the feasible Avatar placement in the first step. However, there is no guarantee that the feasible Avatar placement exists in any scenario. For instance, as shown in Fig. 6, there are 4 UEs and each UE's Avatar is placed in its nearby cloudlet. Assume the capacity of each cloudlet is 3 (each cloudlet can host 3 Avatars) and $\varphi = 0$ (which indicates that if a UE roams from source BS into destination BS, its Avatar should be migrated to the cloudlet associated with the destination BS in order to satisfy the SLA). Then, we consider the case that UE A roams from BS 1 into BS 2, and so UE A's Avatar should migrate to Cloudlet 2 in order to guarantee the SLA. However, Cloudlet 2 cannot host any Avatar without violating its capacity limitation. Thus, obviously, there is no feasible Avatar placement that can satisfy Constraints (9)–(12) simultaneously in this scenario. Therefore, we try to minimize the number of Avatars, which violate the SLA.

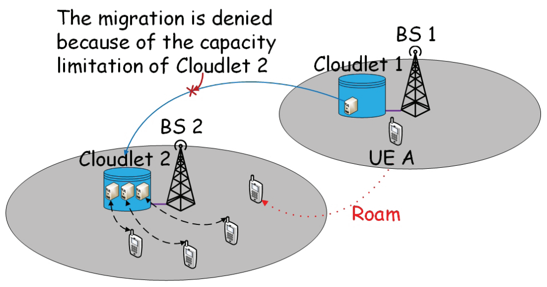

The migration is denied because of the capacity limitation of Cloudlet 2
BS 1
Cloudlet 1
BS 2
Cloudlet 2
UE A
Roam

Fig. 6. The scenario with no feasible Avatar placement.

Definition 1. *If UE i's Avatar is placed in cloudlet k and the E2E delay between UE i and its Avatar does not exceed the SLA φ, we say cloudlet k is UE i's **feasible cloudlet**. Denote $\boldsymbol{\mathcal{K}}_i$ as UE i's **feasible cloudlet set**, which contains all the UE i's feasible cloudlets, i.e., $\boldsymbol{\mathcal{K}}_i = \left\{k | \sum\limits_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk} \leq \varphi\right\}$.*

Based on Def. 1, we define b_{ik} as the profit for placing UE i's Avatar into cloudlet k, i.e.,

$$\forall i \in \boldsymbol{\mathcal{I}}, \forall k \in \boldsymbol{\mathcal{K}}, \begin{cases} b_{ik} = 1, & if\ k \in \boldsymbol{\mathcal{K}}_i, \\ b_{ik} = \frac{1}{|\boldsymbol{\mathcal{I}}|}\frac{\varphi}{\sum\limits_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk}}, & if k \in \boldsymbol{\mathcal{K}} \backslash \boldsymbol{\mathcal{K}}_i, \end{cases} \tag{13}$$

where $|\boldsymbol{\mathcal{I}}|$ is the total number of UEs in the network. Eq. 13 indicates that if UE i's Avatar is placed in one of the UE i's feasible cloudlets (i.e., the SLA is not violated), the GCN provider can obtain one unit profit (i.e., $b_{ik} = 1$). Otherwise, the GCN provider only obtains $\frac{1}{|\boldsymbol{\mathcal{I}}|}\frac{\varphi}{\sum\limits_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk}}$ unit of profit. Note that the SLA is violated by placing UE i's Avatar in cloudlet k for $k \in \boldsymbol{\mathcal{K}} \backslash \boldsymbol{\mathcal{K}}_i$, and so $\frac{\varphi}{\sum\limits_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk}} < 1$. Assume there are a huge number of UEs in GCN (i.e., $\frac{1}{|\boldsymbol{\mathcal{I}}|} \rightarrow 0$), then $b_{ik} \rightarrow 0$ ($k \in \boldsymbol{\mathcal{K}} \backslash \boldsymbol{\mathcal{K}}_i$). Therefore, we can formulate the feasible Avatar placement problem, denoted as $\boldsymbol{P2}$, as follows:

$$\boldsymbol{P2}: \quad \arg\max_{x_{ik}} \sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{k\in\boldsymbol{\mathcal{K}}} b_{ik}x_{ik} \tag{14}$$
$$s.t. \quad Constraints \ (10), (11), (12).$$

The objective of $\boldsymbol{P2}$ is to maximize the total profit of the GCN provider. In order to achieve the goal, UE's Avatars are preferred to be placed in the their feasible cloudlets as much as possible. This is equivalent to the goal of minimizing the number of the Avatars, which violate the SLA.

In order to efficiently solve $\boldsymbol{P2}$, we design a FeasiblE Avatar placemenT (FEAT) algorithm, as shown in Algorithm 1. Specifically, FEAT comprises two parts: in the first part (i.e., *Step* 1 in Algorithm 1), without considering the cloudlet capacity constraint, each UE's Avatar is placed in one of the UE's feasible cloudlets that has the largest residual capacity (we define cloudlet k's residual capacity as $c_k = \epsilon n_k - \sum\limits_{i\in\boldsymbol{\mathcal{I}}} x_{ik}$) to host the Avatars. This is done by balancing the residual capacity among the cloudlets so that the number of cloudlets, whose hosting Avatars exceed their respective capacities, is minimized after all the Avatars have been placed. However, the mentioned Avatar placement is not the feasible solution of $\boldsymbol{P2}$. FEAT needs to adjust the placement to satisfy the capacity constraint for each cloudlet while minimizing the profit reduction.

After all the Avatars are placed in their feasible cloudlets, we separate the cloudlets into two sets, denoted as $\mathcal{Z}_1$ and $\mathcal{Z}_2$. The cloudlets in $\mathcal{Z}_1$ violate their capacity limitations, i.e., $\mathcal{Z}_1 = \left\{k| \sum_{i\in\mathcal{I}} x_{ik} > \epsilon n_k\right\}$, and the cloudlets in $\mathcal{Z}_2$ still have enough space to host the Avatars, i.e., $\mathcal{Z}_2 = \left\{k| \sum_{i\in\mathcal{I}} x_{ik} < \epsilon n_k\right\}$. In the second part (*Steps* $3-8$ in Algorithm 1), FEAT tries to move a suitable Avatar from cloudlet $k \in \mathcal{Z}_1$ to a suitable cloudlet in $\mathcal{Z}_2$ (so that the profit reduction is minimized) in each iteration until none of the cloudlets violate their capacity limitations, i.e., $|\mathcal{Z}_1| = \emptyset$. Specifically, denote $\mathcal{I}^{'}$ as the set of UEs, whose Avatars are placed in the cloudlets in $\mathcal{Z}_1$.

Definition 2. *For each $i \in \mathcal{I}^{'}$, if k_i ($k_i \in \mathcal{Z}_2$) is the cloudlet, which generates the maximum profit b_{ik_i} by hosting UE i's Avatar, i.e., $k_i = \underset{k}{argmax}\left\{b_{ik}|k \in \mathcal{Z}_2\right\}$, we say k_i is UE i's* ***alternative cloudlet*** *and b_{ik_i} is UE i's* ***alternative profit****. Note that if there are many cloudlets in $\mathcal{Z}_2$ that generate the same maximum profit with respect to UE i's Avatar, we will pick the cloudlet with the maximum residual capacity as UE i's alternative cloudlet; if more than one cloudlet has the same maximum profit and the same residual capacity with respect to UE i's Avatar, we will randomly pick the one among those cloudlets as UE i's alternative cloudlet.*

Based on Def. 2, for each UE $i \in \mathcal{I}^{'}$, FEAT can find its alternative cloudlet in $\mathcal{Z}_2$. In order to minimize the profit reduction, FEAT will pick a suitable Avatar, which is originally placed in $\mathcal{Z}_1$, and move it into its alternative cloudlet based on the following definition.

Definition 3. *In order to minimize the profit reduction after the Avatar replacement, FEAT will pick the Avatar, which has the maximum alternative profit (i.e., $i = \arg\max_{i'} \left\{i'|b_{i'k_{i'}}, i' \in \mathcal{I}^{'}\right\}$), and move it into its alternative cloudlet. Note that if more than one Avatar has the same maximum alternative profit, we will pick the Avatar, whose alternative cloudlet has the maximum residual capacity, and move it into its alternative cloudlet; if more than one Avatar has the same maximum alternative profit and their alternative cloudlets have the same residual capacity, we will randomly pick the one among those Avatars and move it into its alternative cloudlet.*

After moving the suitable Avatar, which is selected based on Def. 3, into its alternative cloudlet, two cloudlet sets (i.e., $\mathcal{Z}_1$ and $\mathcal{Z}_2$) and the Avatars' alternative cloudlets will be updated accordingly. FEAT will iteratively move the suitable Avatar into its alternative cloudlet until $|\mathcal{Z}_1| = \emptyset$. The following lemma proves the convergence of the FEAT algorithm:

Lemma 1. *If $|\mathcal{I}| \le \sum_{k\in\mathcal{K}} \epsilon n_k$, FEAT converges and produces a feasible solution of $\boldsymbol{P2}$ after a finite number of iterations.*

Proof: $|\mathcal{I}| \le \sum_{k\in\mathcal{K}} \epsilon n_k$ implies the capacity of the total cloudlets in GCN is no less than the total number of Avatars. Thus, for each UE $i \in \mathcal{I}^{'}$, there always exists an alternative cloudlet $k \in \mathcal{Z}_2$ for UE i so that FEAT can move UE i's Avatar from its original cloudlet (whose number of hosting Avatars exceeds its capacity limitation) into its alternative cloudlet k (which has enough space to host at least one Avatar).

Denote ϑ as the number of *excessive Avatars* (note that the number of *excessive Avatars* equals to the total number of Avatars hosted by the cloudlets in $\mathcal{Z}_1$ minus the total capacity of these cloudlets) generated in the first part of FEAT (i.e., *Step* 1 in Algorithm 1), i.e., $\vartheta = \sum_{k\in\mathcal{Z}_2} (x_{ik} - \epsilon n_k)$, where $\vartheta \le \epsilon n_k$. In each iteration of the second part of FEAT (*Steps* $4-7$ in Algorithm 1), one *excessive Avatar* will be moved into its alternative cloudlet (which is proven to exist if $|\mathcal{I}| \le \sum_{k\in\mathcal{K}} \epsilon n_k$), i.e., $\vartheta = \vartheta - 1$. FEAT will terminate when $\vartheta = 0$, which implies $|\mathcal{Z}_1| = \emptyset$. ■

Note that the complexity of FEAT depends on the number of iterations and the complexity of each iteration (i.e., the complexity of *Steps* $4-7$ in Algorithm 1) in the algorithm. Specifically, $O(|\mathcal{K}|)$ is the complexity of *Step* 4 in Algorithm 1, $O(|\mathcal{I}||\mathcal{K}|)$ is the complexity of *Step* 5, $O(|\mathcal{I}|)$ is the complexity of *Step* 6, and $O(1)$ is the complexity of *Step* 7. Meanwhile, there are at most $|\mathcal{I}|$ iterations, and thus the complexity of FEAT is $|\mathcal{I}|\left(O(|\mathcal{K}|) + O(|\mathcal{I}||\mathcal{K}|) + O(|\mathcal{K}|) + O(1)\right) = O\left(|\mathcal{I}|^2|\mathcal{K}|\right)$.

Algorithm 1 FEAT algorithm

```
Input: 1) The location vector for all the UEs in GCN, i.e.,
    $\mathcal{Y} = \{y_{ij}|i \in \mathcal{I}, j \in \mathcal{J}\}$. 2) The average E2E delay
    vector $\mathcal{T} = \{t_{jk}|j \in \mathcal{J}, k \in \mathcal{K}\}$.
Output: The Avatar placement vector for all the UEs, i.e.,
    $\overline{\mathcal{X}}_i = \{\overline{x}_{ik}|i \in \mathcal{I}, k \in \mathcal{K}\}$.

1: Place each UE into one of its feasible cloudlets.
2: Initialize $\mathcal{Z}_1$ and $\mathcal{Z}_2$.
3: while $|\mathcal{Z}_1| \neq \emptyset$ do
4:     Update $\mathcal{I}^{'}$;
5:     Update the alternative cloudlet and alternative profit
       for each UE $i$'s Avatar ($i \in \mathcal{I}^{'}$) based on Def. 2;
6:     Select a suitable UE $i$'s Avatar ($i \in \mathcal{I}^{'}$) based on Def.
       3 and move it into its alternative cloudlet;
7:     Update $\mathcal{Z}_1$ and $\mathcal{Z}_2$;
8: end while
9: return $\overline{\mathcal{X}}_i$.
```

B. Balancing the energy demands among the cloudlets

By applying the FEAT algorithm, we can find the feasible solution of $\boldsymbol{P2}$, i.e., minimizing the number of Avatars (which violate the SLA) while guaranteeing the capacity limitation of each cloudlet. However, the energy gap (energy demand minus green energy generation) among the cloudlets in GCN is still unbalanced, which may tremendously increase the total energy consumption of GCN. In the second part of our proposed GAP algorithm, we try to adjust the Avatar placement in order to balance the energy gap among the cloudlets, while ensuring the SLA as well as the capacity limitation for each cloudlet.

Denote $\overline{\boldsymbol{\mathcal{X}}} = \{\overline{x}_{ik} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}}\}$ as the Avatar placement vector, which is generated by FEAT. We define two cloudlet sets, denoted as $\boldsymbol{\Omega_1}$ and $\boldsymbol{\Omega_2}$, where cloudlets in $\boldsymbol{\Omega_1}$ lack green energy and need to pull the energy from the power grid to satisfy the their energy demands, i.e., $\boldsymbol{\Omega_1} = \left\{ k | \sum\limits_{i\in\boldsymbol{\mathcal{I}}} \left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i \right) \overline{x}_{ik} - G_k < 0, k \in \boldsymbol{\mathcal{K}} \right\}$, while cloudlets in $\boldsymbol{\Omega_2}$ have superfluous green energy, i.e., $\boldsymbol{\Omega_2} = \left\{ k | \sum\limits_{i\in\boldsymbol{\mathcal{I}}} \left(\frac{p^{idle}}{\epsilon} + \alpha\mu_i \right) \overline{x}_{ik} - G_k > 0, k \in \boldsymbol{\mathcal{K}} \right\}$. The basic idea of GAP is to iteratively select a suitable Avatar, which is placed in a cloudlet in $\boldsymbol{\Omega_1}$, and move it to a suitable cloudlet in $\boldsymbol{\Omega_2}$ in order to reduce the on-grid power consumption while satisfying the SLA as well as the capacity limitation.

Definition 4. *Denote $\boldsymbol{\mathcal{I}}^{''}$ as the set of UEs, whose Avatars are placed in the cloudlets in $\boldsymbol{\Omega_1}$. We define the available cloudlet set for UE i's Avatar (where $i \in \boldsymbol{\mathcal{I}}^{''}$) as the set of cloudlets in $\boldsymbol{\Omega_2}$, each of which has space to host UE i's Avatar, while satisfying the SLA requirement, i.e., $\boldsymbol{a_i} = \left\{ k | \sum\limits_{i'\in\boldsymbol{\mathcal{I}}} \overline{x}_{i'k} + 1 \le \epsilon n_k, \sum\limits_{j\in\boldsymbol{\mathcal{J}}} y_{ij} t_{jk} \le \varphi, k \in \boldsymbol{\Omega_2} \right\}$, where $\boldsymbol{a_i}$ is the **available cloudlet set** for UE i's Avatar and each cloudlet in $\boldsymbol{a_i}$ is the **available cloudlet** for UE i's Avatar.*

In order to reduce the on-grid power consumption, GAP can move UE i's Avatar ($i \in \boldsymbol{\mathcal{I}}^{''}$) from its original cloudlet into one of its available cloudlets in $\boldsymbol{\Omega_2}$. Since there are probably more than one available cloudlets for UE i's Avatar (i.e., $|\boldsymbol{a_i}| > 1$), GAP will select the available cloudlet with the maximum residual green energy to host UE i's Avatar in order to balance the energy gap among cloudlets. We define this available cloudlet, denoted as ς_i, as the **backup cloudlet** for UE i's Avatar, i.e.,

$$\varsigma_i = \underset{k\in\boldsymbol{a_i}}{\operatorname{argmax}} \{G_k - p_k\}, \tag{15}$$

where G_k is the green energy generation of cloudlet k and p_k is the energy demand of cloudlet k, which can be obtained from Eq. 5. Denote Δ_i ($i \in \boldsymbol{\mathcal{I}}^{''}$) as the total amount of on-grid power savings by migrating UE i's Avatar from its original cloudlet into its backup cloudlet ς_i, i.e.:

$$\begin{aligned}\Delta_i = &\sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k) - \max\left\{\sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k - \rho_i), 0\right\} \\ &+ \max\left\{(p_{\varsigma_i} - G_{\varsigma_i} + \rho_i), 0\right\},\end{aligned} \tag{16}$$

where ρ_i is the energy demand of UE i's Avatar, which can be derived based on Eq. 6; $\sum\limits_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k)$ implies the total on-grid power consumption of the cloudlet, in which UE i is originally placed; $\left\{ \max\left[\sum\limits_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k - \rho_i), 0\right] + \max\left[(p_{\varsigma_i} - G_{\varsigma_i} + \rho_i), 0\right] \right\}$ indicates the total on-grid power consumption of the original cloudlet and the backup cloudlet for UE i's Avatar after the migration, where $\max\left\{\sum\limits_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k - \rho_i), 0\right\}$ refers to the on-grid power consumption of the original cloudlet by removing the energy demands of UE i's Avatar and $\max\{(p_{\varsigma_i} - G_{\varsigma_i} + \rho_i), 0\}$ refers to the on-grid power consumption of backup cloudlet ς_i by adding the energy demands of UE i's Avatar. If UE i's Avatar does not have its backup cloudlet (i.e., the available cloudlet set of UE i's Avatar is empty), $\Delta_i = 0$.

In each iteration, GAP would select the Avatar, which can save the maximum on-grid power consumption, and move it into its backup cloudlet. Denote i^* as the selected UE's Avatar in the current iteration, i.e.,

$$i^* = \arg\max_i \left\{ \Delta_i | i \in \boldsymbol{\mathcal{I}}^{''} \right\}. \tag{17}$$

GAP continues to select a suitable Avatar and move it into its backup cloudlet until $\max\left\{\Delta_i | i \in \boldsymbol{\mathcal{I}}^{''}\right\} = 0$, which indicates that GAP cannot reduce the on-grid power consumption by moving any Avatar into its backup cloudlet. GAP is summarized in Algorithm 2. The following lemma proves the convergence of the GAP algorithm.

Lemma 2. *GAP produces a feasible Avatar placement with respect to $\boldsymbol{P0}$ after a finite number of iterations.*

Proof: For each iteration, GAP will select UE i^*'s Avatar (where i^* is determined by Eq. 17) and move it into its backup cloudlet in order to save e units of on-grid power consumption, where $e = \Delta_{i^*} = \max\left\{\Delta_i | i \in \boldsymbol{\mathcal{I}}^{''}\right\}$. For any UE i' Avatar ($i \in \boldsymbol{\mathcal{I}}^{''}$) whose available cloudlet set is not an empty set (i.e., $\boldsymbol{a_i} = \emptyset$), we have

$$\max\left\{\sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k - \rho_i), 0\right\} < \sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k).$$

Thus,

$$\begin{aligned}\Delta_i = &\sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k) - \max\left\{\sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k - \rho_i), 0\right\} \\ &+ \max\left\{(p_{\varsigma_i} - G_{\varsigma_i} + \rho_i), 0\right\} \\ &> \sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k) - \sum_{k\in\boldsymbol{\Omega_1}} \overline{x}_{ik}(p_k - G_k) = 0.\end{aligned}$$

Thus, if $\boldsymbol{a_i} = \emptyset$, $\Delta_i = 0$; otherwise, $\Delta_i > 0$. Consequently, if the available set of any UE i's Avatar ($i \in \boldsymbol{\mathcal{I}}^{''}$) is an empty set, $e = 0$ (note that if $e = \max\left\{\Delta_i | i \in \boldsymbol{\mathcal{I}}^{''}\right\} = 0$, the GAP algorithm is terminated/has converged); otherwise, $e > 0$. In other words, in order to keep the GAP algorithm running, e should be larger than 0.

Consider that if GAP does not converge, i.e., the total number of the iterations $n \to +\infty$, then $\sum\limits_{n=1}^{+\infty} e_n \to +\infty$. Denote E ($E \ll +\infty$) as the total on-grid power consumption for applying the Avatar placement based on the FEAT algorithm; denote E^* ($E \ge E^* \ge 0$) as the optimal on-grid power consumption for solving $\boldsymbol{P0}$. Thus, we have $\sum\limits_n e_n \le E - E^* \le E \ll +\infty$, which contradicts the fact that GAP does not converge (i.e., $\sum_{n=1}^{+\infty} e_n \to +\infty$). Therefore, we conclude that GAP can reduce the total on-grid power consumption in each iteration and converge after finite iterations. ■

Algorithm 2 GAP algorithm

Input: 1) The location vector for all the UEs in GCN, i.e., $\boldsymbol{\mathcal{Y}} = \{y_{ij} | i \in \boldsymbol{\mathcal{I}}, j \in \boldsymbol{\mathcal{J}}\}$. 2) The average E2E delay vector $\boldsymbol{\mathcal{T}} = \{t_{jk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\}$. 3) The average CPU utilization vector for all the UEs' Avatars, i.e., $\boldsymbol{\mathcal{M}} = \{\mu_i | i \in \boldsymbol{\mathcal{I}}\}$. 4) The green energy generation vector for all the cloudlets, i.e., $\boldsymbol{\mathcal{G}} = \{G_k | k \in \boldsymbol{\mathcal{K}}\}$.

Output: The Avatar placement vector for all the UEs, i.e., $\boldsymbol{\mathcal{X}} = \{x_{ik} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}}\}$.

```
1:  $\overline{\boldsymbol{\mathcal{X}}} = FEAT(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{T}})$.    ▷ Execute FEAT.
2:  Initialize $\boldsymbol{\Omega_1}$ and $\boldsymbol{\Omega_2}$.
3:  Initialize $\boldsymbol{\mathcal{I}}''$.
4:  $\forall i \in \boldsymbol{\mathcal{I}}''$, initialize the available cloudlet set $\boldsymbol{a_i}$ for UE $i$' Avatar based on Def. 4.
5:  $\forall i \in \boldsymbol{\mathcal{I}}''$, initialize the backup cloudlet $\varsigma_i$ for UE $i$' Avatar based on Eq. 15.
6:  $\forall i \in \boldsymbol{\mathcal{I}}''$, initialize $\Delta_i$ based on Eq. 16.
7:  while $\max\left\{\Delta_i | i \in \boldsymbol{\mathcal{I}}''\right\} \neq 0$ do
8:      Select UE $i^*$'s Avatar based on Eq. 17 and move it into its backup cloudlet.
9:      Update $\boldsymbol{\mathcal{I}}''$;
10:     $\forall i \in \boldsymbol{\mathcal{I}}''$, update $\boldsymbol{a_i}$, $\varsigma_i$ and $\Delta_i$;
11: end while
12: return $\boldsymbol{\mathcal{X}}$.
```

Note that the complexity of executing $Steps$ $1-6$ in Algorithm 2 is $O\left(|\boldsymbol{\mathcal{I}}|^2 |\boldsymbol{\mathcal{K}}|\right)$ (which is mainly determined by the complexity of FEAT in $Step$ 1). The complexity of executing $Steps$ $7-11$ is $O\left(|\boldsymbol{\mathcal{I}}|^2\right)$. Consequently, the complexity of GAP is $O\left(|\boldsymbol{\mathcal{I}}|^2 |\boldsymbol{\mathcal{K}}|\right) + O\left(|\boldsymbol{\mathcal{I}}|^2\right) = O\left(|\boldsymbol{\mathcal{I}}|^2 |\boldsymbol{\mathcal{K}}|\right)$.

V. Simulation Results

In this section, we first evaluate the performance (i.e., the optimality) of the proposed GAP algorithm in a small-scale network. For comparisons, we will use the commercial solver, i.e., CPLEX, by applying the branch and cut method to generate the solution of $\boldsymbol{P0}$ (i.e., the optimal solution of the Avatar placement problem). Note that the reason for applying a small-scale network to evaluate the performance of GAP is that CPLEX cannot solve $\boldsymbol{P0}$ in a large-scale network because as the number of UEs and BSs increases, the branch and cut tree becomes so large that it requires a huge amount of memory to solve the problem (e.g., over 17,000 GB size of memory for a network with 13,000 UEs and 2360 BSs).

Second, we simulate the GAP algorithm in a large-scale network in order to demonstrate its scalability and how much power it can save. For comparisons, we show the total on-grid power consumption incurred by other two Avatar placement methods, i.e., FAR and SAP. The basic ideas of FAR and SAP have been explained in the Introduction section.

A. *Performance of the GAP algorithm*

We will first investigate the optimality of the GAP algorithm in a small-scale network topology, which is shown in Fig. 7 with 16 cloudlet-BS combinations (4×4) in a square area of 64 km^2. The coverage of each BS is a square area of 4 km^2. The whole area is divided into 2 parts, i.e., urban and rural areas. Initially, there are 1000 UEs uniformly distributed in the network and each UE's Avatar is placed in its nearest feasible cloudlet (which has enough capacity to host the Avatar). UE mobility adopts the modified random waypoint model, i.e., each UE randomly selects a speed between 0 and 10 m/s in every time slot and moves toward its destination, and the locations of UEs destinations (i.e., the values of x and y coordinates) are randomly selected according to a normal distribution $N(4km, 2km)$, which implies that UEs more likely move toward the center of each urban area (i.e., based on the characteristics of the normal distribution, UEs more likely select their destinations which are close to the center of the network). Moreover, in order to guarantee the E2E delay between a UE and its Avatar in meeting the SLA, we assume that the UE's Avatar can only be placed in the cloudlet, whose connected BS is a neighbor of the UE's BS. For instance, as shown in Fig. 7, if one UE is located in the coverage area of BS 1, its Avatar can only be placed in BS 2, BS 3, BS 4 or BS 5's cloudlet in order to satisfy the SLA requirement.

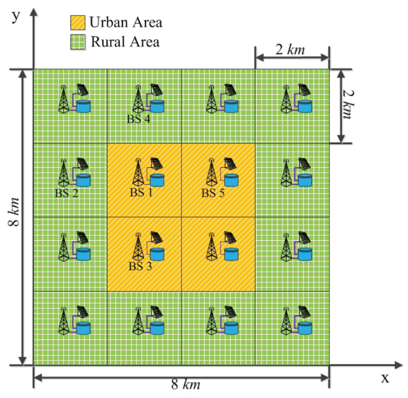


Fig. 7. Network topology.

The resource capacity of each Avatar is homogeneous; each Avatar is configured with 2-core CPU, 4GB memory, and 500 Mbps bandwidth. The Google cluster data trace [45] is applied to emulate the CPU utilization of each Avatar. Specifically, we select the machines with CPU and memory capacity of 0.5 (normalized) in the Google cluster data trace, and calculate their CPU utilization in each time slot; then, the resource utilizations of Avatars are emulated to be the same as those of the machines. The capacity of each cloudlet is 20 homogeneous PMs and each PM can host at most 5 Avatars.

For the daily green energy generation (from 6 am to 6 pm), we use the local solar radiation[3] data trace (Millbrook, NY in Apr. 30th, 2015) from National Climatic Data Center

[3]Green energy generation (G) = Solar radiation (S) $\times$ Solar pannel size (ι) $\times$ Efficiency (η)

[46]. As shown in Fig. 8, the trace collects the solar radiation generation of different locations in the area in each hour. The figure does exhibit the spatial and temporal dynamics of green energy generation, i.e., the amount of solar radiation varies among different locations in the area (especially when the solar radiation generation is relatively large) and the amount of solar radiation in the same location varies over time. In the simulation, we randomly select each cloudlet's solar radiation between S_{max} and S_{min} at time slot t, where S_{max} and S_{min} equal to the values of max and min solar radiation generation at time slot t in Fig. 8, respectively. The other parameters are shown in Table I.

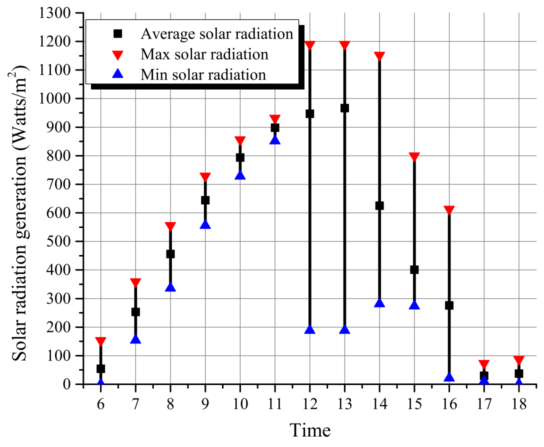


Fig. 8. The daily solar radiation trace.

TABLE I
SYSTEM PARAMETERS

Parameter	Value
Power consumption of the PM in idle mode, p^{idle}	135 $Watts$
CPU uutilization to power mapping coefficient, α	0.65 $Watts/\%$
The efficiency of converting solar rediation to power, η	0.2
Solar panel size, ι	15 m^2

Lemma 3. ***P*0** *is a Mixed Integer Linear Programming (MILP) problem.*

Proof: Denote s_k as the on-grid power consumption of cloudlet k, i.e., $s_k = \max\left\{\sum_{i\in\mathcal{I}}\left(\frac{p^{idle}}{\epsilon}+\alpha\mu_i\right)x_{ik}-G_k, 0\right\}$. Thus, $\boldsymbol{P0}$ is equivalent to:

$$\boldsymbol{P3}: \arg\min_{s_k, x_{ik}} \sum_{k\in\mathcal{K}} s_k \tag{18}$$

$$s.t.\quad \forall k\in\mathcal{K},\quad s_k \geq \sum_{i\in\mathcal{I}}\left(\frac{p^{idle}}{\epsilon}+\alpha\mu_i\right)x_{ik}-G_k, \tag{19}$$

$$\forall k\in\mathcal{K},\quad s_k\geq 0, \tag{20}$$

$$and\ Constraints\ (9),(10),(11),(12).$$

where x_{ik} is a binary variable (the placement of UE i's Avatar) and s_k is a continuous variable indicating the on-grid power consumption of cloudlet k. Obviously, $\boldsymbol{P3}$ is a MILP problem and can be solved by applying the $cplexmilp()$ function in the CLPEX solver during the simulations. ■

We simulate GAP during the day (from 6 am to 6 pm) and the performance of GAP is shown in Fig. 9. We can see that the on-grid power consumption in each time slot incurred by GAP is quite similar to that incurred by CLPEX. We further calculate the total on-grid power consumption during the day by adopting GAP and CLPEX. As shown in Fig. 10, GAP only consumes about 12.8% more on-grid power during the day as compared to the result generated by CPLEX. Yet, CLPEX consumes 52.7% more average execution time in each time slot during the day as compared to that consumed by GAP. Thus, we conclude that the performance of GAP is close to CLPEX, but GAP is much more computationally efficient.

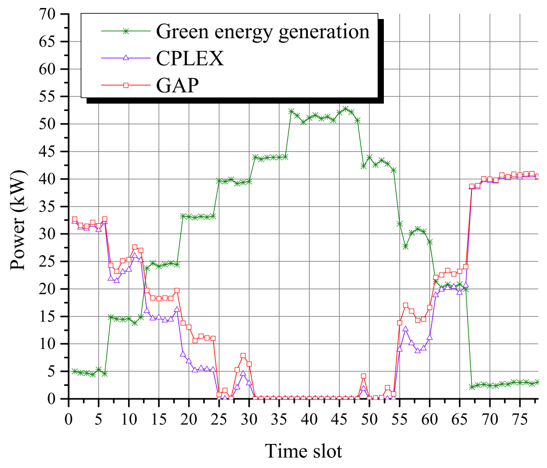


Fig. 9. The performance of GAP during the day.

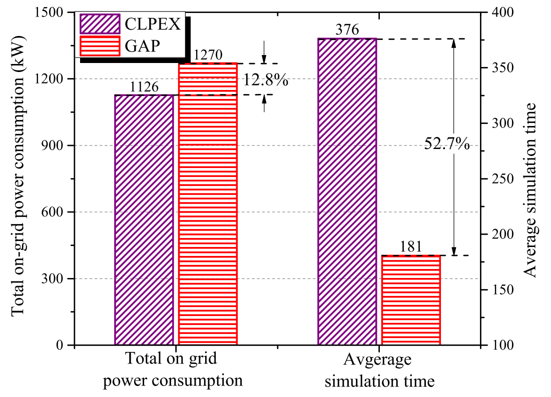


Fig. 10. The total on-grid power consumption and the average execution time.

B. Energy saving of GAP in a large-scale network

In this section, we will demonstrate the scalability and the economic gain (i.e., the amount of energy saving) of GAP as compared to the other two Avatar placement strategies, namely, FAR and SAP, in a large-scale cloudlet network. As mentioned previously, CPLEX requires tremendous memory resources (over 17,000 GB) to solve the problem in a large-scale network topology. This makes CPLEX infeasible to solve the problem in a real network.

We have obtained data traces of more than 13,000 UEs collected from an operating mobile network and extracted their mobility in one day. The whole area contains 2,360 BSs and each UE's location (i.e., the UE within BS's coverage area) is monitored for each ten minutes during the day (from 6 am to 6 pm). Suppose all the UEs' mobility can be accurately predicted, and we use this UE mobility trace as input parameters (i.e., the values of $\mathcal{Y}$) to show the performance of the algorithms. Fig. 11 shows the maximum, the minimum and the variance of the number of local UEs for a BS (i.e., the number of UEs within the coverage area of a BS) among all the BSs in GCN in each time slot. Interestingly, the variance of the number of local UEs among BSs remains stable over time because different UEs' mobilities exhibit similar trends, i.e., most of the users move from their homes to workplaces in the morning, and return back from working places to their homes at night. Thus, these periodical movements result in the stability of the UE density variance among BSs over time. Fig. 11 also shows the spatial dynamics of the UE density among BSs during each time slot, i.e., the energy demands among different cloudlets vary in each time slot if each cloudlet serves its local UEs' Avatars.

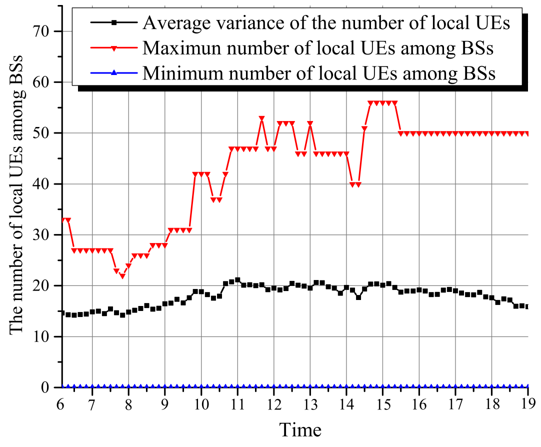


Fig. 11. The statistical results of the UE mobility trace.

Each BS is attached with a cloudlet and the solar radiation generation for each cloudlet during the day also follows the data from National Climatic Data Center. Meanwhile, we still adopt the Google data trace to emulate the CPU utilization of each Avatar in each time slot during the day. Since the average UE density per BS in the large-scale network is smaller than that in the small-scale network, we reduce the capacity of each cloudlet, i.e., 6 PMs are deployed in each cloudlet and each cloudlet can host at most 5 Avatars; meanwhile, the size of solar panel equipped in each cloudlet is 2 m^2 and the E2E delay bound between a UE and its Avatar (i.e., SLA) is 25 ms. The rest of the simulation parameters are the same as those in the small-scale network.

We calculate the total on-grid power consumption in the cloudlet network in each time slot and the results are shown in Fig 12. Intuitively, there is a big on-grid power consumption gap between GAP and SAP/FAR because GAP can migrate the energy demands (i.e., the Avatars) from the cloudlets with no residual green energy to the cloudlets with sufficient residual green energy so that the green energy can be fully utilized. As shown in Fig. 13, the total on-grid power consumptions of GAP, SAP and FAR during the day are 6423.1 kW, 14952.2 kW and 15141.8 kW, respectively, i.e., GAP can save 57.1% and 57.6% of on-grid power consumption as compared to SAP and FAR, respectively. Note that the on-grid power consumption of SAP and FAR are quite similar. This can be explained by the similar trends among different UEs as well, i.e., during the morning, UEs are mostly located at home, and so the on-grid power consumption of SAP and FAR are mainly generated by the cloudlets located in the residential areas; during the working hours, UEs are mostly located in the working places, and so the on-grid power consumption of FAR is mainly attributed to the cloudlets located in the working places (note that the on-grid power consumption of SAP is still mainly generated by the cloudlets in the residential areas because the location of each Avatar is static after it is initially placed). This kind of energy demands shifting between the residential areas and the working places incurs the similar on-grid power consumption by applying SAP and FAR. We further test the average SLA violation rate[4] during the day by applying GAP, SAP and FAR, and the results are shown in Fig. 13. Clearly, GAP and FAR can guarantee that the E2E delay between a UE and its Avatar is no longer than the SLA, which is 25 ms. In SAP, all the Avatars' locations are static, and so the SLA may be violated if some UEs roam further away. Consequently, on average, 41% of Avatars violate the SLA in each time slot. Although SAP consumes the similar on-grid power consumption as FAR and generates the highest SLA violation rate, SAP does not introduce extra Avatar migrations, which may consume extra power and increase the traffic load of the SDN-based cellular core [47], [48]. In the future study, we will establish a migration cost model and incorporate it into the Avatar placement strategy.

We further test the total on-grid power consumption in the cloudlet network by increasing the number of UEs. The mobility trace of each additional UE is the same as that of the existing UE, which is randomly selected among the existing UEs. As shown in Fig. 14, the total amount of energy saving (between GAP and FAR/SAP) increases as the total number of UEs increases because as the number of UEs increases, more cloudlets tend to lack green energy or the cloudlets (which already lack green energy) will consume more on-grid power, and thus GAP can potentially reduce the on-grid power consumption by migrating the Avatars among the cloudlets.

We also analyze how the value of SLA affects the performance of GAP. As shown in Fig. 15, when the value of SLA increases, the total on-grid power decreases accordingly because as the value of SLA increases, each Avatar will have more available cloudlets, which can increase the chance of a cloudlet (which has higher energy demands and lacks green energy) to migrate its Avatars into other cloudlets (which have sufficient green energy) in order to reduce its on-grid power consumption without violating the SLA and the cloudlet capacity constraint. For instance, assume that there is a UE in GCN and its Avatar can only be placed in Cloudlet-A in order to satisfy the SLA (e.g., 20 ms), i.e., Cloudlet A is

[4]SLA violation rate=the number of Avatars violate the SLA ÷ the total number of Avatars

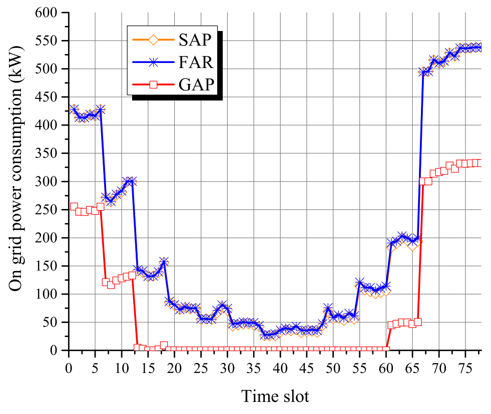


Fig. 12. The on-grid power consumption of the cloudlet network in each time slot.

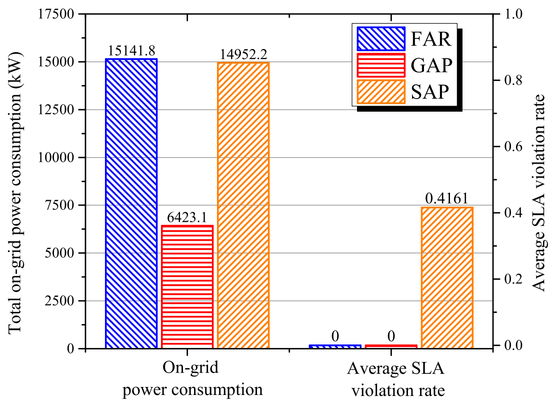


Fig. 13. The total on-grid power consumption of the cloudlet network and the average SLA violation rate during the day.

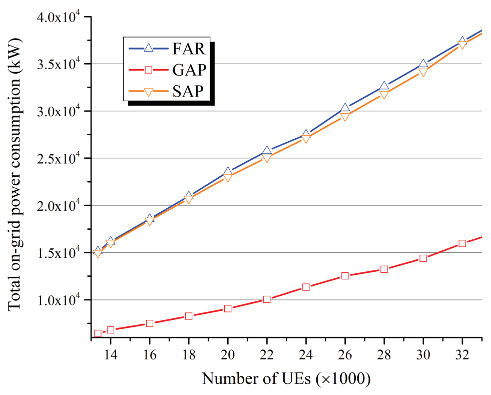


Fig. 14. The total on-grid power consumption of the cloudlet network during the day over different number of UEs.

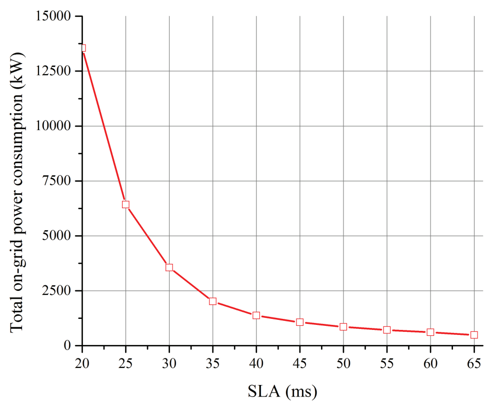


Fig. 15. The total on-grid power consumption of the cloudlet network during the day over different values of SLA.

the only available cloudlet of the UE's Avatar. If the value of the SLA increases (e.g., 25 ms), Cloudlet-B becomes the available cloudlet of the UE's Avatar as well, i.e., the Avatar can be placed in either Cloudlet-A or Cloudlet-B to satisfy the SLA. Consequently, the Avatar can be placed in Cloudlet-B to further reduce the on-grid power consumption if Cloudlet-A does not have residual green energy but Cloudlet-B has. This indicates that a larger value of the SLA facilitates GAP to further reduce the total on-grid power consumption.

VI. Conclusion

In this paper, we have proposed the GCN architecture to facilitate big data networking as well as MCC applications. Specifically, each UE can access its own Avatar, considered as the UE's private computing resources, with the low E2E delay. In order to reduce the operational cost for maintaining the distributed cloudlets, each cloudlet is powered by both green and brown energy. Fully utilizing green energy can significantly reduce the operational cost of cloudlet providers. However, owing to the spatial dynamics of energy demand and green energy generation, some cloudlets' energy demands can be fully provided by green energy but others need to utilize on-grid power to satisfy their energy demands. In order to minimize the total on-grid power consumption of GCN, we have proposed the GAP algorithm to distribute the energy demands by migrating the Avatars among cloudlets according to the cloudlets' residual green energy, while satisfying the SLA and cloudlet capacity constraints. We have demonstrated via simulations that the performance of GAP is compatible to that of CPLEX, but with a much lower computational complexity. By applying the data traces extracted from the real world, we have demonstrated the scalability and the economic gain of GAP. Spefically, as compared to the other two Avatar placement strategies, i.e., SAP and FAR, GAP can save 57.1% and 57.6% of on-grid power consumption, respectively, while satisfying the SLA. Meanwhile, as the number of UEs in GCN increases, GAP can save more on-grid power consumption.

References

[1] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 46–53, June 2013.

[2] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani and R. Buyya, "Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 337-368, First Quarter 2014.

[3] X. Sun, N. Ansari and R. Wang, "Optimizing resource utilization of a data center," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2822–2846, Fourth Quarter, 2016.

[4] Y. Zhang and N. Ansari,"On architecture design, congestion notification, TCP incast and power consumption in data centers," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 39–64, First Quarter 2013.

[5] S.R. Ellis, K. Mania, B.D. Adelstein, and M.I. Hill, "Generalizeability of latency detection in a variety of virtual environments," *Proc. Human Factors and Ergonomics Soc. Annual Meeting*, New Orleans, LA, Sep. 20–24, 2004, vol. 48, no. 23, pp. 2632–2636.

[6] K. Ha, *et al.*, *Adaptive VM handoff across cloudlets*, Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.

[7] X. Sun and N. Ansari, "Green cloudlet network: a distributed green Mobile cloud network," *IEEE Netw.*, vol. 31, no. 1, pp. 64–70, Jan./Feb. 2017.

[8] M. Whaiduzzaman, A. Naveed, and A. Gani, "MobiCoRE: mobile device based cloudlet resource enhancement for optimal task response," *IEEE Trans. Services Comput.*, doi: 10.1109/TSC.2016.2564407, early access.

[9] M. Chen, *et al.*, "Privacy protection and intrusion avoidance for cloudlet-based medical data sharing," *IEEE Trans. Cloud Comput.*, doi: 10.1109/TCC.2016.2617382, early access.

[10] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, July 2017.

[11] N. Ansari and X. Sun, "Mobile edge computing empowers internet of things," *IEICE Trans. Commun.*, doi: 10.1587/transcom.2017NRI0001, early access.

[12] C. Borcea, X. Ding, N. Gehani, R. Curtmola, M. A. Khan and H. Debnath, "Avatar: mobile distributed computing in the cloud," *2015 3rd IEEE Intl. Conf. Mobile Cloud Comput., Serv., and Eng.*, San Francisco, CA, 2015, pp. 151–156.

[13] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient customization of mobile computing infrastructure," *Proc. First Wksp. Virtualization Mobile Comput.*, Breckenridge, CO, Jun. 17–20, 2008, pp. 37–41.

[14] X. Sun and N. Ansari, "EdgeIoT: mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22-29, Dec. 2016.

[15] X. Sun and N. Ansari, "Cloudlet networks: empowering mobile networks with computing capabilities," *IEEE COMSOC MMTC Commun.-Frontiers*, vol. 12, no. 4, pp. 6-11, July 2017.

[16] X. Sun and N. Ansari,"PRIMAL: PRofIt Maximization Avatar pLacement for Mobile Edge Computing," *Proc. IEEE Intl. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 23–27, 2016, pp. 1–6.

[17] X. Jin, L.E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," *Proc. the ninth ACM conf. Emerging Netw. Experiments Technol.*, Santa Barbara, CA, Dec. 09–12, 2013, pp. 163–174.

[18] A. Lara, A. Kolasani and B. Ramamurthy, "Network innovation using OpenFlow: a survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, First Quarter 2014.

[19] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep.–Oct. 2013.

[20] T. Han and N. Ansari, "On optimizing green energy utilization for cellular networks with hybrid energy supplies," *IEEE Trans. Wireless Commun.*, vol. 12, no. 8, pp. 3872–3882, Aug. 2013.

[21] T. Han and N. Ansari, "On greening cellular networks via multicell cooperation," *IEEE Wireless Commun.*, vol. 20, no. 1, pp. 82–89, Feb. 2013.

[22] X. Sun, N. Ansari and Q. Fan, "Green energy aware Avatar migration strategy in green cloudlet networks," *2015 IEEE 7th Intl. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Vancouver, BC, 2015, pp. 139–146.

[23] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" *IEEE Intl. Conf. Comp. Commun. (INFOCOM)*, Toronto, ON, 2014, pp. 1060–1068.

[24] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[25] E. Cuervo, *et al.*, "MAUI: making smartphones last longer with code offload," *Proc. the 8th Intl. Conf. Mobile Syst., Appl., and Serv.*, San Francisco, CA, Jun. 15–18, 2010, pp. 49–62.

[26] B.G. Chun, *et al.*, "Clonecloud: elastic execution between mobile device and cloud," *Proc. of the sixth Conf. Comp. Syst.*, Salzburg, Austria, Apr. 10–13, 2011, pp. 301–314.

[27] S. Kosta, *et al.*, "ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading," *2012 Proc. IEEE INFOCOM*, Orlando, FL, 2012, pp. 945–953.

[28] D.T. Hoang, D. Niyato and L. B. Le, "Simulation-based optimization for admission control of mobile cloudlets," *2014 IEEE Intl. Conf. Commun.*, Sydney, NSW, 2014, pp. 3764–3769.

[29] X. Sun and N. Ansari, "Avaptive Avatar handoff in the cloudlet network," *IEEE Trans. Cloud Comput.*, doi: 10.1109/TCC.2017.2701794, early access.

[30] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," *Proc. 2008 Conf. Power aware Comput. and Syst.*, San Diego, CA, Dec. 7, 2008, pp. 3–7.

[31] X. Zhang, J.J. Lu, X. Qin, and X.N. Zhao, "A high-level energy consumption model for heterogeneous data centers," *Simulation Modelling Practice and Theory*, vol. 39, pp. 41–55.

[32] E. Cho, S.A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," *Proc. the 17th ACM SIGKDD Intl. Conf. Knowl. Discovery and Data Mining*, San Diego, CA, Aug. 21–24, 2011, pp. 1082–1090.

[33] W. Su, S.J. Lee, and M. Gerla. "Mobility prediction in wireless networks," *21st Century Mil. Commun. Conf. Proc.(MILCOM)*, Los Angeles, CA, Oct. 22–25, 2000, pp. 491–495.

[34] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 6, pp. 2577–2590, June 2015.

[35] N.L.M. Adrichem, C. Doerr, and F. Kuipers, "OpenNetMon: network monitoring in openflow software-defined networks," *2014 IEEE Netw. Operations and Manag. Symposium (NOMS)*, Krakow, Poland, May. 05–09, pp. 1–8.

[36] C. Yu, *et al.*, "Software-defined latency monitoring in data center networks," *Passive and Active Measurement*, vol. 8995, pp. 360-372, Mar., 2015.

[37] M. Musolesi, "Big mobile data mining: good or evil?," *IEEE Internet Comput.*, vol. 18, no. 1, pp. 78–81, Jan.–Feb. 2014.

[38] J. Liu, Y. Liang, and N. Ansari, "Spark-based large-scale matrix inversion for big data processing," *IEEE Access*, vol. 4, pp. 2166-2176, 2016.

[39] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[40] M. Isard, *et al.*, "Dryad: distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 3, pp. 59–72, 2007.

[41] N. Quang-Hung, *et al.*, "A genetic algorithm for power-aware virtual machine allocation in private cloud," *Inform. and Commun. Technol.-EurAsia Conf.*, Yogyakarta, Indonesia, Mar. 25–29, 2013, pp. 183–191.

[42] A. Beloglazov and R. Buyya. "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," *2010 ACM/IFIP/USENIX Intl. Middleware Conf.*, Bangalore, India, Nov. 29–Dec. 3, 2010, pp. 4–9.

[43] R. Nathuji and K. Schwan, "Virtualpower: coordinated power management in virtualized enterprise systems," *ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 6, pp. 265–278, October 2007.

[44] U. Deshpande and K. Keahey, "Traffic-sensitive live migration of virtual machines," *2015 15th IEEE/ACM Intl. Symposium on Cluster, Cloud and Grid Comput.*, Shenzhen, China, 2015, pp. 51-60.

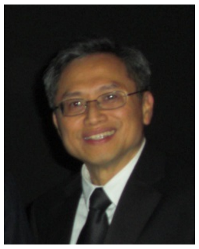
[45] Google Cluster Data. [Online]. Available: https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md.

[46] Daily solar radiation data trace from National Climatic Data Center. [Online]. Available: http://www1.ncdc.noaa.gov/pub/data/uscrn/products/hourly02/2015/CRNH0203-2015-NY_Millbrook_3_W.txt

[47] H. Liu, H. Jin, C.Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Comput.*, vol. 16, no. 2, pp. 249–264, 2013.

[48] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?" *2013 IEEE 27th Intl. Conf. Advanced Inform. Netw. and Appl. (AINA)*, Barcelona, Spain, Mar. 25–28, 2013, pp. 514–521.

Xiang Sun [S'13] received a B.E. degree in electronic and information engineering and an M.E. degree in technology of computer applications from Hebei University of Engineering, Hebei, China. He is currently working towards the Ph.D. degree in electrical engineering at the New Jersey Institute of Technology (NJIT), Newark, New Jersey. His research interests include mobile edge computing, big data networking, green computing and communications, content/resource caching in Internet of Things, and drone-aided mobile access networks.



Nirwan Ansari [S'78, M'83 ,SM'94, F'09] is Distinguished Professor of Electrical and Computer Engineering at the New Jersey Institute of Technology (NJIT). He has also been a visiting (chair) professor at several universities such as High-level Visiting Scientist at Beijing University of Posts and Telecommunications.

He recently authored Green Mobile Networks: A Networking Perspective (Wiley-IEEE, 2017) with T. Han, and co-authored two other books. He has also (co-)authored more than 500 technical publications, over 200 in widely cited journals/magazines. He has guest-edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, and various aspects of broadband networks.

He was elected to serve in the IEEE Communications Society (ComSoc) Board of Governors as a member-at-large, has chaired ComSoc technical committees, and has been actively organizing numerous IEEE International Conferences/Symposia/Workshops. He has frequently delivered keynote addresses, distinguished lectures, tutorials, and invited talks. Some of his recognitions include IEEE Fellow, several Excellence in Teaching Awards, some best paper awards, the NCE Excellence in Research Award, the IEEE TCGCC Distinguished Technical Achievement Recognition Award, the ComSoc AHSN TC Technical Recognition Award, the NJ Inventors Hall of Fame Inventor of the Year Award, the Thomas Alva Edison Patent Award, Purdue University Outstanding Electrical and Computer Engineer Award, and designation as a COMSOC Distinguished Lecturer. He has also been granted 35 U.S. patents.

He received a Ph.D. from Purdue University in 1988, an MSEE from the University of Michigan in 1983, and a BSEE (summa cum laude with a perfect GPA) from NJIT in 1982.