Adaptive Avatar Handoff in the Cloudlet Network

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

In a traditional big data network, data streams generated by User Equipments (UEs) are uploaded to the remote cloud (for further processing) via the Internet. However, moving a huge amount of data via the Internet may lead to a long End-to-End (E2E) delay between a UE and its computing resources (in the remote cloud) as well as severe traffic jams in the Internet. To overcome this drawback, we propose a cloudlet network to bring the computing and storage resources from the cloud to the mobile edge. Each base station is attached to one cloudlet and each UE is associated with its Avatar in the cloudlet to process its data locally. Thus, the E2E delay between a UE and its computing resources in its Avatars is reduced as compared to that in the traditional big data network. However, in order to maintain the low E2E delay when UEs roam away, it is necessary to hand off Avatars accordingly—it is not practical to hand off the Avatars' virtual disks during roaming as this will incur unbearable migration time and network congestion. We propose the LatEncy Aware Replica placemeNt (LEARN) algorithm to place a number of replicas of each Avatar's virtual disk into suitable cloudlets. Thus, the Avatar can be handed off among its cloudlets (which contain one of its replicas) without migrating its virtual disk. Simulations demonstrate that LEARN reduces the average E2E delay. Meanwhile, by considering the capacity limitation of each cloudlet, we propose the LatEncy aware Avatar hanDoff (LEAD) algorithm to place UEs' Avatars among the cloudlets such that the average E2E delay is minimized. Simulations demonstrate that LEAD maintains the low average E2E delay.

Index Terms—**cloudlet, mobile edge computing, big data, mobile cloud computing, Avatar, handoff, replica.**

I. Introduction

Portable User Equipments (UEs), such as smart phones, tablets, smart watches and smart glasses, come with a rich set of embedded sensors already built-in [1]. By utilizing these intelligent UEs, the location (e.g., GPS information), activity (e.g., walking, speaking, sitting, etc.), mood (e.g., happy, calm, alert, etc.), health information (e.g., blood pressure, heartbeat rate, body temperature, etc.), and the ambient environment information of each human being can be monitored and recorded. Thus, UEs are considered as data stream generators producing massive amounts of data, and analyzing these data is not only extremely valuable for market applications, but also has an incredible potential to benefit society as a whole [2]. For instance, analyzing the videos and photos captured by UEs is crucial to localize lost children or terrorists, and analyzing the users' activities, possible events and historical traffic statistics can accurately forecast the traffic condition to help drivers selecting optimal driving directions. However, the value of the big data decreases as time passes by (for instance, it is important to identify the terrorists from the recent photos/videos quickly). Therefore, analyzing the big data in real-time is critical. In a traditional big data network, all the data generated by UEs are transmitted to a data center (for further analysis) via the Internet [3]–[5] because the data center can provision efficient distributed computing architecture (such as MapReduce [6], Dryad [7], and Storm [8]) as well as flexible resource allocation [9]. However, transmitting the big data from UEs to the data center through the Internet leads to long network latency and drains the network resources. Thus, the existing big data networking platform is not suitable for real-time big data analysis.

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.

Rather than bringing the data to the computing resources, it is more efficient to bring the computing resources to the data in order to reduce the network latency and the traffic load of the network. Thus, we propose a new cloudlet network architecture to analyze UEs' data streams at the mobile edge. As shown in Fig. 1, each Base Station (BS) is attached to a cloudlet, which can be considered as a distributed tiny data center that is deployed close to UEs. Each UE is associated with a specific Avatar, i.e., a dedicated Virtual Machine (VM) providing private computing, communications and storage resources to the UE, in the nearby cloudlet. Thus, the data streams generated from UEs can be uploaded and analyzed in their own Avatars with low End-to-End (E2E) delay. On the top of the cloudlets, Software Defined Network (SDN) based cellular core network [10]–[13] has been introduced in the cloudlet network architecture to provide efficient and flexible communications paths between Avatars in different cloudlets as well as between UEs in different BSs. Moreover, every UE and its Avatar in the cloudlet can communicate with public data centers (e.g., Amazon EC2) and Storage Area Networks (SANs) via the Internet in order to provision scalability, i.e., if cloudlets are not available for UEs because of the capacity limitation, UEs' Avatars can be migrated to the remote data centers to continue serving their UEs. The communications between a UE and a BS is orchestrated by the Packet Data Convergence Protocol [14], which is specified by 3GPP. The communications protocol between a BS and an OpenFlow switch as well as between a BS and a cloudlet adopts the TCP/IP protocol.

The proposed cloudlet network architecture facilitates the real-time big data analysis. A typical example by utilizing the cloudlet network architecture to analyze the big data is the terrorist localization application, which is to identify and track terrorists by analyzing the photos and videos taken by different UEs. Specifically, each UE uploads its captured photos and videos to its Avatar. The terrorist localization application sends the terrorists' photos to Avatars, which compare terrorists'

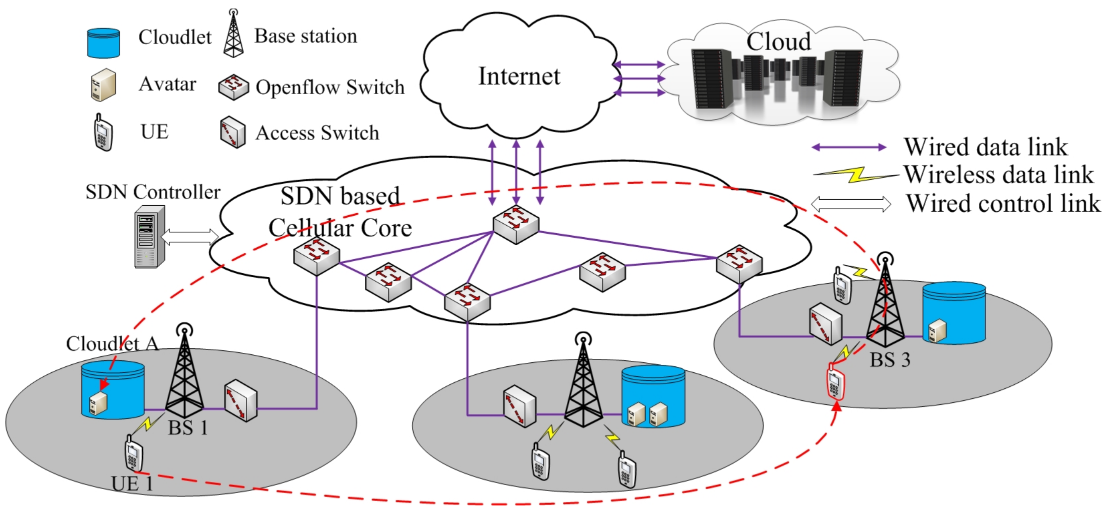


Fig. 1. The cloudlet network architecture.

photos to the captured photos and videos by running face matching algorithms. If matched, the information (e.g., locations and timestamps) of the photos and videos would be transmitted to the terrorist localization application for further processing. Applying the cloudlet network architecture to implement terrorist localization has the following advantages. First, Avatars are considered as private VMs for their UEs. Avatars locally analyze the data streams generated from their UEs and share metadata (rather than raw data) by removing UEs' personal information from raw data. For instance, in the terrorist localization application, each Avatar only provides the locations and the time stamps of the matched photos and videos rather than the photos and videos to the terrorist localization application. This can preserve the privacy of each UE. Second, as compared to the traditional method (in which all the captured photos and videos would be uploaded to a data center for analysis), analyzing each UE's captured photos and videos in its Avatar (which is placed in the local cloudlet) can significantly reduce the traffic load of the network and network delay for uploading the photos and videos to the computing resources, thus speeding up the process of terrorist localization.

In order to encourage UEs in subscribing their own Avatars in the cloudlet network, each Avatar can be considered as a software clone [17] of its UE so that the UE can automatically offload some computational intensive tasks to its Avatar by utilizing Mobile Cloud Computing (MCC) technologies [10], [15], [16]. Avatars execute the tasks and return back the results to their UEs, and so the task execution time and the energy consumption of UEs can be reduced significantly [18]. Note that many MCC frameworks have been designed to enable UEs to offload their resource-hungry tasks to the cloud. For example, the MAUI project [19] provides method level code offloading based on the .NET framework. Three mobile applications, i.e., a face recognition application, an arcade game application, and a voice-based language translation application, have been tested under MAUI and the results demonstrated by conducting code offloading can significantly reduce the energy consumption of mobile devices and improve the performance in terms of the delay of these mobile applications. The CloudClone project [17] designs an MCC framework by migrating an application thread from the mobile device at a chosen point to the application VM (which is considered as the application-level clone of the mobile device) in the cloud. Virus scanning, image search, and behavior profiling applications have been tested under the designed MCC framework to demonstrate the reduction of the mobile device's energy consumption and the application's execution time by offloading application workloads from mobile devices to their clones. By applying the mentioned MCC frameworks, it is feasible and beneficial to enable UEs to outsource their computational intensive tasks to their Avatars in the nearby cloudlet. Therefore, Avatars play two roles in the cloudlet network, i.e., the big data analyzer and the MCC application outsourcer.

The rest of the paper is organized as follows. In Section II, we propose to hand off Avatars among cloudlets in order to maintain the low E2E delay between UEs and their Avatars when UEs roam away. In order to avoid virtual disk migration during the Avatar handoff process, we propose to place a number of replicas of the Avatar's virtual disk among suitable cloudlets. In Section III, we formulate the Avatar replica placement problem and design the LatEncy Aware Replica placemeNt (LEARN) algorithm to solve the problem. In Section IV, by considering the capacity limitation of each cloudlet, we propose the LatEncy aware Avatar hanDoff (LEAD) algorithm to optimally place UEs' Avatars among the cloudlets in each time slot such that the average E2E delay between UEs and Avatars is minimized. In Section V, we demonstrate the performance of the proposed LEARN and LEAD algorithm via extensive simulations. In Section VI, we briefly review the related works. The conclusion is presented in Section VII.

II. Avatar Handoff

UEs are roaming among BSs over time and so the E2E delay between UEs and their Avatars may become worse if the Avatars remain in their original cloudlets. For instance, as shown in Fig. 1, if UE 1 roams from BS 1's coverage area into BS 3's coverage area and its Avatar still resides in Cloudlet A, the communications path between UE 1 and its Avatar should traverse the SDN based cellular core, which may increase the E2E delay as well as the traffic load of the SDN based cellular core. 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 to determine the overall E2E delay when UE roams away. Thus, we refer to the E2E delay between a UE and its Avatar as the E2E delay between the BS (which serves the UE) and the cloudlet (which contains the UE's Avatar) in the rest of the paper.

Long E2E delay can significantly degrade the performance of the big data analysis as well as the MCC applications. Obviously, spending less time for uploading the data streams to Avatars will benefit real-time big data analysis to produce more valuable results. Meanwhile, the E2E delay is critical for MCC applications. It is reported that augmented reality applications require an E2E delay of less than 16 ms [20] and the cloud-based virtual desktop applications require an E2E delay of less than 60 ms [21]. Thus, it is critical to preserve the low E2E delay between a UE and its Avatar by migrating the Avatar among cloudlets when the UE roams away. An Avatar is considered as a private VM, which comprises isolated vCPUs, memory, and virtual disks, and so migrating an Avatar between cloudlets is to conduct live VM migration [22] between cloudlets over the SDN based cellular core. We refer to the Avatar migration process as the Avatar handoff in the rest of the paper.

The Avatar handoff process needs to migrate the whole Avatar (which includes the memory, the virtual disk[1], and the dirty blocks of the memory and the virtual disk generated during the migration process) from the source into the destination cloudlet. The total Avatar handoff time determines the performance of the Avatar handoff [10]. This is because, first, the degraded E2E delay between a UE and its Avatar persists until the Avatar handoff process is finished, and so shorter handoff time will produce lower E2E delay; second, the Avatar handoff process consumes extra resources of the Avatar (especially the bandwidth resource), and thus degrades the performance of applications currently running in the Avatar. Therefore, short handoff time will improve the performance of the applications. However, it is reported that migrating the whole Avatar between two cloudlets over a network with stable 10 $Mbps$ bandwidth and 50 ms Round Trip Time (RTT) consumes over two hours [21]; this indicates that handing off a whole Avatar between cloudlets cannot maintain the low E2E delay between the Avatar and its UE but exhausts the resource of the network and the Avatar. The main reason for incurring the unacceptable handoff time is to migrate the large volume of the Avatar's virtual disk over the network [21].

In order to avoid virtual disk migration during the handoff process, we propose to place a number of replicas of an Avatar's virtual disk in the suitable cloudlets. The replicas of an Avatar[2] are synchronized with the Avatar's virtual disk during a fixed time period (e.g., 5 min). Thus, if a UE's Avatar tries to hand off to the cloudlet which contains one of the Avatar's replicas, only the memory and the *pre-handoff virtual disk dirty blocks* of the Avatar are needed to be transmitted to the destination cloudlet. The *pre-handoff virtual disk dirty blocks* of the Avatar means the virtual disk dirty blocks that are generated after the last replica synchronization process. For instance, as shown in Fig. 2, the Avatar's replicas are synchronized at t_1; meanwhile, the Avatar handoff is triggered at t_2, and thus the number of the virtual disk blocks, which are modified during the interval between t_1 and t_2, are defined as the *pre-handoff virtual disk dirty blocks*, which need to be migrated during the handoff process. Since the dirty block generation rate of the virtual disk is relatively low, only very small portion of the virtual disk is needed to be transmitted to the destination cloudlet, which will significantly reduce the handoff time.

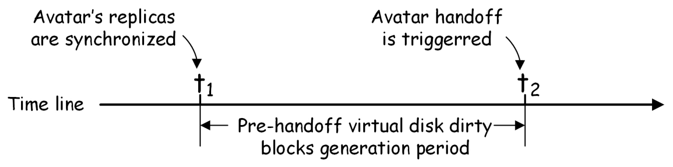

Avatar's replicas are synchronized
Avatar handoff is triggerred
t_1
t_2
Time line
Pre-handoff virtual disk dirty blocks generation period

Fig. 2. The illustration of pre-handoff virtual disk dirty blocks.

III. Avatar replica placement

Migrating the whole virtual disk of an Avatar during the Avatar handoff process incurs unbearable handoff time and increases the traffic load of the SDN-based cellular core significantly. Thus, in order to avoid the virtual disk migration during the Avatar handoff process, we pre-deploy a number of the Avatar's replicas among the cloudlets. A cloudlet, which contains one of an Avatar's replicas, is defined as the Avatar's *available cloudlet*. Thus, an Avatar can only be migrated to its *available cloudlets*.

Note that it is unnecessary and inefficient to place the Avatar's replicas in all the cloudlets in the network because increasing the number of replicas for each Avatar increases the capital expenditure (CAPEX) of the cloudlet provider (by implementing more storage space in the cloudlets) as well as the synchronization traffic in the SDN based cellular core. Meanwhile, placing the Avatar's replicas in the cloudlets, which are never visited by the UE, cannot benefit the communications between the UE and its Avatar. Therefore, it is important to optimally place a limited number of replicas for each Avatar among the cloudlets so that the average E2E delay (during a period ΔT, e.g., one day) between the UE and its

[1]In order to guarantee the performance of the big data analysis and the MCC applications running in the Avatar, the whole virtual disk of the Avatar should be located in the same physical machine with its vCPU and memory providing low I/O latency.

[2]The replicas of an Avatar are referred to as the replicas of the Avatar's virtual disk in the rest of the paper.

Avatar can be minimized (by utilizing Avatar handoff) when the UE roams in the network.

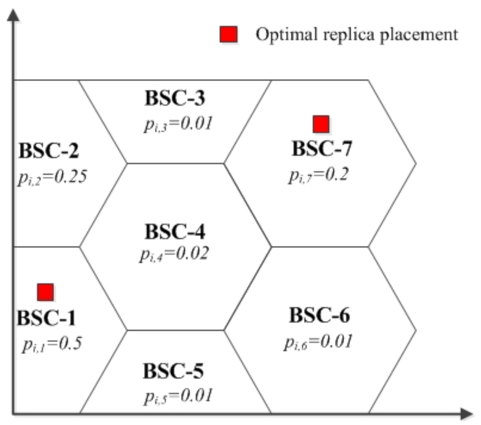


Fig. 3. The illustration of the optimal Avatar replica placement.

Normally, the Avatar's replicas will be placed where its UE will commonly visit (it has been demonstrated that about 10% to 30% of all human movement can be explained by their social relationship, while 50% to 70% is attributed to periodic behaviors [23]; thus, we believe that the dynamics of future human movement can be reliably predicted based on the mathematical models [23]–[25]), such as home and workplace. However, this is not the optimal Avatar replica placement strategy. For instance, suppose the cloudlet network topology is shown in Fig. 3, which contains 7 BS-Cloudlet (BSC) combinations[3], and two replicas of UE i's Avatar need to be placed. Meanwhile, suppose the occurrence probability of UE i in BS j's coverage area, denoted as p_{ij}[4] (where $j = 1, 2, \cdots, 7$), is also shown in Fig. 3. Thus, traditionally, two replicas will be placed in BSC-1 and BSC-2 (because $p_{i,1}$ and $p_{i,2}$ are the two largest values, implying that UE i will most commonly visit BSC-1 and BSC-2). Yet, deploying the two replicas in BSC-1 and BSC-7 may be the optimal solution for UE i. This is because, first, the value of $p_{i,2}$ and $p_{i,7}$ are close; second, BSC-1 and BSC-2 are adjacent to each other, and so the E2E delay between UE i and its Avatar is low even if UE i is in the BSC-2's coverage area and its Avatar is in BSC-1. On the contrary, since BSC-7 is far away from BSC-1, the E2E delay may be unbearable if UE i is in the BSC-7's coverage area and its Avatar is in BSC-1, and thus placing the 2nd replica in BSC-7 may improve the average E2E delay significantly.

Therefore, we conclude that the value of p_{ij} is not the only determinant to affect the performance of the Avatar replica placement. The E2E delay between different BSCs can also affect the performance of Avatar replica placement.

A. System model

Let $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ be the set of UEs, BSs and cloudlets, respectively. Denote x_{ik} as a binary variable indicating one replica of 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, let t_{jk} be the average E2E delay between BS j and cloudlet k. The value of t_{jk} ($j \neq k$) can be measured and recorded by the SDN controller [26], [27]. Note that if $j = k$, we say that cloudlet k is BS j's attached cloudlet. Moreover, denote y_{ijk} as a binary variable indicating UE i's Avatar is located in cloudlet k (i.e., $y_{ijk} = 1$) or not (i.e., $y_{ijk} = 0$) when UE i is in BS j's coverage area. Let τ_{ij} be the average E2E delay between UE i and its Avatar when UE i is in the BS j's coverage area, then we have:

$$\tau_{ij} = \sum_{k \in \mathcal{K}} t_{jk} y_{ijk}. \tag{1}$$

Denote τ_i as the average E2E delay between UE i and its Avatar during the period ΔT (e.g., one day); meanwhile, let p_{ij} be the predicted occurrence probability of UE i in BS j's coverage area during the period ΔT; then, we have:

$$\tau_i = \sum_{j \in \mathcal{J}} p_{ij} \tau_{ij} = \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} p_{ij} t_{jk} y_{ijk}. \tag{2}$$

The optimal Avatar replica placement for each UE i ($i \in \mathcal{I}$) is to minimize its average E2E τ_i during the period ΔT. Thus, we formulate the problem as follows:

$$\boldsymbol{P0}: \quad \underset{x_{ik},\ y_{ijk}}{\arg\min} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} p_{ij} t_{jk} y_{ijk} \tag{3}$$

$$s.t. \sum_{k \in \mathcal{K}} x_{ik} = \kappa, \tag{4}$$

$$\forall j \in \mathcal{J}, \ \sum_{k \in \mathcal{K}} y_{ijk} = 1, \tag{5}$$

$$\forall j \in \mathcal{J} \ \forall k \in \mathcal{K}, \ y_{ijk} \leq x_{ik}, \tag{6}$$

$$\forall k \in \mathcal{K}, \ x_{ik} \in \{0, 1\}, \tag{7}$$

$$\forall j \in \mathcal{J} \ \forall k \in \mathcal{K}, \ y_{ijk} \in \{0, 1\}, \tag{8}$$

where κ is the total number of replicas that can be deployed among the cloudlets for each UE's Avatar. Constraint (4) requires that exactly κ replicas are placed for UE i. Constraint (5) indicates that UE i' Avatar should be located in exactly one cloudlet when UE i is in BS j's coverage area. Constraint (6) implies that UE i' Avatar can only be located in a cloudlet if and only if the cloudlet contains one of its replicas (i.e., in order to satisfy Constraint (6), y_{ijk} could equal to 1 iff $x_{ik} = 1$; otherwise, y_{ijk} should be 0 if $x_{ik} = 0$). Constraints (7) and (8) implies x_{ik} and y_{ijk} ($j \in \mathcal{J}$ and $k \in \mathcal{K}$) are binary variables.

Lemma 1. *The Avatar replica placement problem (i.e., $\boldsymbol{P0}$) is NP-hard when $\kappa > 1$.*

Proof: The formulation of the Avatar replica placement problem is equivalent to the p-median problem [28] where $\kappa = p > 1$, and the p-median problem has been proved to be NP-hard on a general network topology (note that it has been demonstrated that the p-median problem can be solved in polynomial time $O(n^2p^2)$ only if the network is a tree [29]). Therefore, we need to demonstrate the topology of the proposed cloudlet network is not a tree.

[3]A BSC combination indicates that a BS is attached to a dedicated cloudlet.
[4]$p_{ij} = \frac{\textit{the total time that UE i stays in the BS j's coverage area}}{\textit{the total time period (i.e., one day)}}$

Based on the cloudlet network proposed in Sec. I, each BS can communicate with all the cloudlets over the SDN based cellular core. Thus, the cloudlet network can be considered as a complete graph in which every vertex represents the BS-cloudlet pair. Every pair of distinct vertices is connected by a unique edge, which represents a communications link with a dedicated cost in terms of the E2E delay. Therefore, the Avatar replica placement problem is NP-hard. ■

B. LatEncy Aware Replica placemeNt (LEARN)

Inspired by the Lagrangian relaxation algorithm for solving the p-median problem [30], we design the LatEncy Aware Replica placemeNt (LEARN) algorithm to optimally place the replicas among cloudlets for each UE. The basic idea of LEARN is to iteratively obtain the lower bound (LB) and upper bound (UB) of the Avatar replica placement problem through Lagrangian procedure until the differece between the LB and UB is less than a predefined value ψ.

Specifically, we relax Constraint (5) in $\boldsymbol{P0}$ to obtain the following Lagrangian problem:

$\boldsymbol{P1}$:

$$\begin{aligned}\max_{\lambda_j}\min_{x_{ik}y_{ijk}}\mathcal{L}&=\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}p_{ij}t_{jk}y_{ijk}+\sum_{j\in\mathcal{J}}\lambda_j\left(1-\sum_{k\in\mathcal{K}}y_{ijk}\right)\\&=\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\left(p_{ij}t_{jk}-\lambda_j\right)y_{ijk}+\sum_{j\in\mathcal{J}}\lambda_j, \qquad (9)\\ &s.t.\quad Constraints\ (4),(6),(7),(8),\end{aligned}$$

where λ_j ($\forall j\in\mathcal{J},\lambda_j\geq 0$) are the Lagrangian multipliers. For fixed values of the Lagrange multipliers λ_j, the above relaxed problem (i.e., $\boldsymbol{P1}$) will yield an optimal objective value that is an LB on any feasible solution of the original Avatar replica placement problem (i.e., $\boldsymbol{P0}$).

Lemma 2. *Define vector $\boldsymbol{\Delta}_i=\{\Delta_{ik}|k\in\mathcal{K}\}$, where $\Delta_{ik}=\sum_{j\in\mathcal{J}}\min\left(0,p_{ij}t_{jk}-\lambda_j\right)$; define the cloudlet set $\mathcal{K}_i^{'}$ ($\mathcal{K}_i^{'}\subset\mathcal{K}$), where $\left|\mathcal{K}_i^{'}\right|=\kappa$ and $\left\{\Delta_{ik}|k\in\mathcal{K}_i^{'}\right\}$ are the κ number of the smallest values in vector $\boldsymbol{\Delta}_i$. Then, for any given set of multipliers $\boldsymbol{\lambda}=\{\lambda_j|j\in\mathcal{J}\}$, the optimal solution of the Lagrangian problem, denoted as $\boldsymbol{\mathcal{X}}_i^*=\{x_{ik}^*|k\in\mathcal{K}\}$ and $\boldsymbol{\mathcal{Y}}_i^*=\left\{y_{ijk}^*|j\in\mathcal{J},k\in\mathcal{K}\right\}$, can be expressed as follows:*

$$\forall k\in\mathcal{K},\ \ x_{ik}^*=\begin{cases}1, & k\in\mathcal{K}_i^{'}.\\0, & otherwise.\end{cases} \qquad (10)$$

$$\forall j\in\mathcal{J},\forall k\in\mathcal{K},y_{ijk}^*=\begin{cases}1, & p_{ij}t_{jk}-\lambda_j<0\ \&\ x_{ik}^*=1.\\0, & otherwise.\end{cases} \qquad (11)$$

Proof: Obviously, in order to minimize the objective function of the Lagrangian problem (i.e., $\boldsymbol{P1}$), y_{ijk} should be chosen its maximum value if $p_{ij}t_{jk}-\lambda_j\leq 0$ ($j\in\mathcal{J},k\in\mathcal{K}$) for any given set of Lagrangian multipliers $\boldsymbol{\lambda}=\{\lambda_j|j\in\mathcal{J}\}$, otherwise, $y_{ijk}=0$. Thus, by considering Constraint (6), the optimal solution of y_{ijk}^* is given by: $y_{ijk}^*=x_{ik}$, if $p_{ij}t_{jk}-\lambda_j\leq 0$; $y_{ijk}^*=0$, otherwise.

By substituting the optimal solution of y_{ijk}^* into $\mathcal{L}$ (Eq. (9)), the Lagrangian problem is transformed into:

$$\begin{aligned}\min_{x_{ik}}\mathcal{L}&=\sum_{k\in\mathcal{K}}\Delta_{ik}x_{ik}+\sum_{j\in\mathcal{J}}\lambda_j \qquad (12)\\ &s.t.\quad Constraints\ (4),(7),\end{aligned}$$

where $\Delta_{ik}=\sum_{j\in\mathcal{J}}\min\left(0,p_{ij}t_{jk}-\lambda_j\right)$. For any given set of Lagrangian multipliers $\boldsymbol{\lambda}$, the above problem is trivial to solve, i.e., $x_{ik}^*=1$, if $k\in\mathcal{K}_i^{'}$ (where $\mathcal{K}_i^{'}$ is the set of κ number of the cloudlets, which have the smallest values of Δ_{ik} in $\boldsymbol{\Delta}_i=\{\Delta_{ik}|k\in\mathcal{K}\}$); $x_{ik}^*=0$, otherwise. Thus, Eq. (10) and Eq. (11) have been proved. ■

Note that solving the relaxed problem can provide the LB of the original Avatar replica problem, i.e.,

$$LB=\sum_{k\in\mathcal{K}}\Delta_{ik}x_{ik}^*+\sum_{j\in\mathcal{J}}\lambda_j. \qquad (13)$$

However, the solution of the Lagrangian relaxation problem may not be the feasible solution with respect to the original Avatar replica problem ($\boldsymbol{P0}$), i.e., Constraint (5) may not be satisfied for the solution $\boldsymbol{\mathcal{Y}}_i^*=\left\{y_{ijk}^*|j\in\mathcal{J},k\in\mathcal{K}\right\}$. In order to obtain a feasible solution of the original problem, denoted as $\overline{\boldsymbol{\mathcal{Y}}}_i=\left\{\overline{y}_{ijk}|j\in\mathcal{J},k\in\mathcal{K}\right\}$, we can simply allocate the Avatar of UE i to the cloudlet, which has the lowest E2E delay among the cloudlets containing one replica of the Avatar, when UE i is in BS j, i.e., for each $j\in\mathcal{J}$, we have:

$$\forall k\in\mathcal{K},\ \overline{y}_{ijk}=\begin{cases}1, & t_{jk}=\min\left\{t_{jk}|k\in\mathcal{K}_i^{''}\right\}\\0, & otherwise.\end{cases} \qquad (14)$$

where $\mathcal{K}_i^{''}$ is the set of available cloudlets (which contain one replica of UE i's Avatar) of UE i's Avatar, i.e., $\mathcal{K}_i^{''}=\{k|x_{ik}^*=1,k\in\mathcal{K}\}$.

Substituting the feasible solution (i.e., $\overline{\boldsymbol{\mathcal{Y}}}_i=\left\{\overline{y}_{ijk}|j\in\mathcal{J},k\in\mathcal{K}\right\}$) into the objective function of the original problem (i.e., Eq. (3)), we have the UB of the original problem:

$$UB=\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}p_{ij}t_{jk}\overline{y}_{ijk}. \qquad (15)$$

Note that the original problem always chooses its UB as its objective value because the UB can guarantee the existence of the feasible solution. However, selecting different values of Lagrange multiplier vector (i.e., $\boldsymbol{\lambda}$) may generate different values of the UB. Thus, by applying the subgradient method [31], we adjust the values of Lagrange multipliers in each iteration in order to obtain the smaller value of UB. The iteration terminates until $UB^{opt}-LB\leq\psi$, where UB^{opt} indicates the best (i.e., smallest) value of UB that has been found in the previous iterations.

In the n^{th} iteration ($n>1$), the values of the Lagrangian multipliers λ_j^n ($j\in\mathcal{J}$) are calculated based on the following expression:

$$\forall j\in\mathcal{J},\lambda_j^n=\max\left\{0,\lambda_j^{n-1}-\theta^n\left\{\sum_{k\in\mathcal{K}}{y_{ijk}^*}^{n-1}-1\right\}\right\}, \qquad (16)$$

where λ_j^{n-1} are the Lagrangian multipliers generated in the previous iteration; ${y_{ijk}^*}^{n-1}$ ($j \in \mathcal{J}, k \in \mathcal{K}$) are the optimal solution of $\boldsymbol{P1}$ (i.e., the relaxed problem) in the previous iteration, which can be calculated based on Eq. (11) and θ^n is the step length adopted in the n^{th} iteration, which can be calculate based on the following expression [32]:

$$\theta^n = \frac{\alpha\left(UB^{opt} - LB^{n-1}\right)}{\sum\limits_{j\in\mathcal{J}}\left(\sum\limits_{k\in\mathcal{K}} {y_{ijk}^*}^{n-1} - 1\right)^2}, \tag{17}$$

where α ($0 < \alpha < 2$) is a decreasing adaptation parameter and LB^{n-1} is the value of LB in the previous iteration (i.e., $LB^{n-1} = \sum\limits_{k\in\mathcal{K}} \Delta_{ik} {x_{ik}^*}^{n-1} + \sum\limits_{j\in\mathcal{J}} \lambda_j^{n-1}$). The detail of the LEARN algorithm is shown in Algorithm 1.

C. One example to illustrate the LEARN algorithm

Suppose there are three BSs in the network and each BS is attached to one cloudlet. Assume the average E2E delay vector is $\mathcal{T} = \begin{bmatrix} 0 & 20 & 15 \\ 20 & 0 & 10 \\ 15 & 10 & 0 \end{bmatrix}$. There is a UE in the network and the occurrence probability of the UE in the respective BSs during the day is $\mathcal{P}$= $[0.5, 0.3, 0.2]$. If we need to place two replicas (i.e., $\kappa = 2$) for its Avatar's virtual disk, then LEARN shall apply the following procedure to obtain the optimal replica placement for the UE:

- Steps 1-2 in Algorithm 1: randomly select the initial values of Lagrangian multipliers, e.g., $\boldsymbol{\lambda}$= $[5, 5, 5]$; initialize $LB = 0$ and $UB^{opt} = +\infty$;
- Steps 4-5 in Algorithm 1: given the value of $\boldsymbol{\lambda}$, calculate the values of $\mathcal{X}^*$ and $\mathcal{Y}^*$ for the UE based on Lemma 2; in this example, $\mathcal{X}^*$= $[1, 1, 0]$ and $\mathcal{Y}^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$. Then, update the value of LB based on Eq. (13); in this example, $LB = 0$;
- Steps 6-7 in Algorithm 1: calculate the value of $\overline{\mathcal{Y}}$ based on Eq. (14). In this example, $\overline{\mathcal{Y}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$. Then, update the value of UB based on Eq. (15); in this example, $UB = 3.5$;
- Steps 8-11 in Algorithm 1: compare UB with UB^{opt}; in this example, $UB < UB^{opt}$, and thus $\mathcal{X}^{opt} = \mathcal{X}^* = [1, 1, 0]$ and $UB^{opt} = UB = 3.5$;
- Steps 12-13 in Algorithm 1: update the value of Lagrangian multipliers, i.e., $\boldsymbol{\lambda}$, based on Eq. (16), and goes back to Steps 4-5 until $UB^{opt} - LB \le \psi$.

The LEARN algorithm is executed offline, i.e., for a fixed period ΔT, LEARN will update the replica placement for different UEs during the off peak hours. Also, the replica placement updating period ΔT can also vary among different UEs. For instance, if UEs have similar behaviors during the workdays, LEARN only needs to update the replica placement of the UEs during the weekends; otherwise, it is preferred to update the replica placement daily. It is worth to note that a centralized controller (or a control function running in the SDN controller) is used to predict the occurrence probability for each UE, obtain the average E2E delay vector among different cloudlets and BSs from the SDN controller, and generate the replica placement vector for each UE by executing the proposed LEARN algorithm.

Algorithm 1 LEARN algorithm

Input: 1) The occurrence probability vector for UE i among BSs, i.e., $\mathcal{P}_i = \{p_{ij} | j \in \mathcal{J}\}$. 2) The average E2E delay vector $\mathcal{T} = \{t_{jk} | j \in \mathcal{J}, k \in \mathcal{K}\}$.

Output: The replica placement vector for UE i, i.e., $\mathcal{X}_i^{opt} = \{x_{ik}^{opt} | k \in \mathcal{K}\}$.

```
1:  Initialize the set of Lagrangian multipliers $\boldsymbol{\lambda} = \{\lambda_j | j \in \mathcal{J}\}$.
2:  Initialize $LB = 0$ and $UB^{opt} = +\infty$.
3:  while $UB^{opt} - LB > \psi$ do
4:      Calculate $\mathcal{X}_i^*$ and $\mathcal{Y}_i^*$ based on Lemma 2;
5:      Update the value of $LB$ based on Eq. (13);
6:      Calculate $\overline{\mathcal{Y}}_i$ based on Eq. (14);
7:      Calculate the value of $UB$ based on Eq. (15);
8:      if $UB < UB^{opt}$ then
9:          $\mathcal{X}_i^{opt} = \mathcal{X}_i^*$;
10:         $UB^{opt} = UB$;
11:     end if
12:     Update step length $\theta^n$ based on Eq. (17);
13:     Update Lagrangian multipliers $\boldsymbol{\lambda}$ based on Eq. (16);
14: end while
15: return $\mathcal{X}_i^{opt}$.
```

IV. Adaptive Avatar handoff

After the replicas of each UE's Avatar being deployed among cloudlets, the Avatar can be handed off among its available cloudlets based on its UE's location. Optimally, the Avatar will be handed off to the available cloudlet, which incurs the lowest E2E among its available cloudlets, when the UE roams into a new location. However, each cloudlet has its CPU and memory capacity, and so the Avatar may not be handed off to the optimal cloudlet because the optimal cloudlet may not have enough residual capacity to host the Avatar. Therefore, it is necessary to design an adaptive Avatar handoff strategy to determine the location of each UE's Avatar in each time slot in order to minimize the average E2E delay between all the UEs and their Avatars during the time slot by jointly considering the capacity limitation of each cloudlet.

Note that different from the Avatar replica placement problem (which tries to generate the replica placement solution for each UE's Avatar based on the statistics for a long time period (e.g., one day)), the adaptive Avatar handoff problem tries to obtain the location of each UE's Avatar (rather the Avatar's replicas) based on real time information (e.g., the current locations of all the UEs) and the problem should be solved in real time.

A. Problem formulation

Let l_{ij} be a binary indicator to identify UE i in BS j's coverage area (i.e., $l_{i,j} = 1$) or not ($l_{i,j} = 0$) in the current time slot. Meanwhile, let z_{ik} be a binary variable to indicate whether UE i's Avatar is in cloudlet k ($z_{ik}=1$) or not ($z_{ik}=0$) in the current time slot. $\boldsymbol{\mathcal{X}}_i^{opt}=\left\{x_{ik}^{opt}|k\in\boldsymbol{\mathcal{K}}\right\}$, which is generated by the LEARN algorithm, is the optimal replica placement vector for UE i. In order to avoid the virtual disk migration, UE i's Avatar can only be allocated to its available cloudlet k (i.e., z_{ik} could equal to 1 iff $k\in\boldsymbol{\mathcal{K}}_i^{''}$, where $\boldsymbol{\mathcal{K}}_i^{''}=\left\{k|x_{ik}^{opt}=1,k\in\boldsymbol{\mathcal{K}}\right\}$). Thus, the average E2E delay between UE i and its Avatar in the current time slot, i.e., τ_i, can be expressed as follows:

$$\tau_i=\sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}_i^{''}}l_{ij}t_{jk}z_{ik}. \tag{18}$$

Suppose all the UEs' Avatars are homogeneous, i.e., all the Avatars have the same CPU, memory and bandwidth configurations. Denote q_k as the capacity of cloudlet k, i.e., the total number of Avatars can be hosted by cloudlet k. Thus, we formulate the Avatar handoff problem as follows:

$$\boldsymbol{P2}:\quad \arg\min_{z_{ik}}\sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}_i^{''}}l_{ij}t_{jk}z_{ik} \tag{19}$$

$$s.t.\quad \forall i\in\boldsymbol{\mathcal{I}},\ \sum_{k\in\boldsymbol{\mathcal{K}}_i^{''}}z_{ik}=1, \tag{20}$$

$$\forall k\in\boldsymbol{\mathcal{K}},\ \sum_{i\in\boldsymbol{\mathcal{I}}}z_{ik}\le q_k, \tag{21}$$

$$\forall i\in\boldsymbol{\mathcal{I}}\ \forall k\in\boldsymbol{\mathcal{K}},\quad z_{ik}\in\{0,1\}, \tag{22}$$

where the objective is to minimize the average E2E delay between all the UEs and their Avatars in the current time slot. Constraint (20) indicates each Avatar should be hosted by one cloudlet, which contains one replica of the UE's Avatar; Constraint (21) implies that each cloudlet has its capacity limitation; Constraint (22) indicates z_{ik} is a binary variable.

Note that it is not easy to solve $\boldsymbol{P2}$ since the available cloudlet set $\boldsymbol{\mathcal{K}}_i^{''}$ varies among different UEs' Avatars that makes the summation index k in the object function of $\boldsymbol{P2}$ to vary among different UEs' Avatars. The following lemma facilitates a dual problem of $\boldsymbol{P2}$, which can be readily solved because the summation index k in the objective function of the new problem does not vary among different UEs' Avatars.

Lemma 3. *Let $\tau_i^{'}$ be*

$$\tau_i^{'}=\sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}}\frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon}z_{ik}, \tag{23}$$

where ε is a very small positive value close to zero. Then, $\boldsymbol{P2}$ can be equivalently transformed into:

$$\boldsymbol{P3}:\quad \arg\min_{z_{ik}}\sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}}\frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon}z_{ik},$$

$$s.t.\quad \forall i\in\boldsymbol{\mathcal{I}},\ \sum_{k\in\boldsymbol{\mathcal{K}}}z_{ik}=1,$$

$$\forall k\in\boldsymbol{\mathcal{K}},\ \sum_{i\in\boldsymbol{\mathcal{I}}}z_{ik}\le q_k,$$

$$\forall i\in\boldsymbol{\mathcal{I}}\ \forall k\in\boldsymbol{\mathcal{K}},\quad z_{ik}\in\{0,1\}.$$

Proof: For each $i\in\boldsymbol{\mathcal{I}}$, if $x_{ik}^{opt}=0$ (i.e., $k\in\boldsymbol{\mathcal{K}}\backslash\boldsymbol{\mathcal{K}}_i^{''}$), $\frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon}\to+\infty$, and thus z_{ik} should be set to zero in order to minimize the value of $\tau_i^{'}$; if $x_{ik}^{opt}=1$ (i.e., $k\in\boldsymbol{\mathcal{K}}_i^{''}$), the expression of $\tau_i^{'}$ is approximately equal to τ_i. Thus, $\boldsymbol{P2}$ and $\boldsymbol{P3}$ are equivalent. ■

By applying Lemma 3, the problem (i.e., $\boldsymbol{P2}$) can be transformed into:

$$\arg\min_{z_{ik}}\sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{k\in\boldsymbol{\mathcal{K}}}c_{ik}z_{ik} \tag{24}$$

$$s.t.\ \ Constraints\ (20),(21),(22),$$

where $c_{ik}=\frac{\sum_{j\in\boldsymbol{\mathcal{J}}}l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon}$ is the weighted E2E delay between UE i and its Avatar located in cloudlet k.

The proposed Avatar handoff problem is considered as a special case of the generalized assignment problem (in which Constraint (21) is depicted as $\forall k\in\boldsymbol{\mathcal{K}}$, $\sum_{i\in\boldsymbol{\mathcal{I}}}a_{ik}z_{ik}\le q_k$, where $a_{ik}>0$), which is proven to be NP-hard [1]. Recently, many heuristics have been designed to find the suboptimal solution of the generalized assignment problem and each of them has its tradeoff between the complexity and the performance. In the proposed network, we need to allocate tens of thousands of UEs into tens of thousands of cloudlets in each time slot. Thus, we design a novel LatEncy aware Avatar hanDoff (LEAD) algorithm to efficiently solve the proposed Avatar handoff problem.

B. LatEncy aware Avatar hanDoff (LEAD)

First, we build a relaxed Avatar handoff problem (i.e., $\boldsymbol{P4}$) by relaxing Constraint (21), i.e.:

$$\boldsymbol{P4}:\quad \arg\min_{z_{ik}}\sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{k\in\boldsymbol{\mathcal{K}}}c_{ik}z_{ik}$$

$$s.t.\quad \forall i\in\boldsymbol{\mathcal{I}},\ \sum_{k\in\boldsymbol{\mathcal{K}}}z_{ik}=1,$$

$$\forall i\in\boldsymbol{\mathcal{I}}\ \forall k\in\boldsymbol{\mathcal{K}},\quad z_{ik}\in\{0,1\}.$$

The optimal solution of $\boldsymbol{P4}$, denoted as $\boldsymbol{\mathcal{Z}}_i^{'}=\left\{z_{ik}^{'}|k\in\boldsymbol{\mathcal{K}}\right\}$, can be easily derived, i.e.:

$$\forall i\in\boldsymbol{\mathcal{I}},\quad z_{ik}^{'}=\begin{cases}1, & k=\arg\min_{\overline{k}}\left\{c_{i\overline{k}}|\overline{k}\in\boldsymbol{\mathcal{K}}\right\}.\\ 0, & otherwise.\end{cases} \tag{25}$$

The optimal solution of $\boldsymbol{P4}$ generates the optimal cloudlet (which incurs the minimum weighted E2E delay among the cloudlets) for each Avatar. However, it may not be the feasible solution of the original Avatar handoff problem, i.e., the total number of Avatars that are hosted by some cloudlets may exceed their capacity. Denote these set of cloudlets as $\boldsymbol{\mathcal{K}}_1$, i.e., $\boldsymbol{\mathcal{K}}_1=\left\{k|\sum_{i\in\boldsymbol{\mathcal{I}}}z_{ik}^{'}>q_k,k\in\boldsymbol{\mathcal{K}}\right\}$; denote the set of cloudlets, which have enough resources to host at least one Avatar, as $\boldsymbol{\mathcal{K}}_2$, i.e., $\boldsymbol{\mathcal{K}}_2=\left\{k|\sum_{i\in\boldsymbol{\mathcal{I}}}z_{ik}^{'}<q_k,k\in\boldsymbol{\mathcal{K}}\right\}$; denote the set of UEs, whose Avatars are hosted by cloudlet $k\in\boldsymbol{\mathcal{K}}_1$, as $\boldsymbol{\mathcal{I}}_1$, i.e., $\boldsymbol{\mathcal{I}}_1=\left\{i|z_{ik}^{'}=1,k\in\boldsymbol{\mathcal{K}}_1,i\in\boldsymbol{\mathcal{I}}\right\}$. The basic idea of the LEAD algorithm is to choose a suitable UE i's Avatar, whose optimal cloudlet has violated its capacity limitation (i.e., $i\in\boldsymbol{\mathcal{I}}_1$), and reallocate the Avatar into its suboptimal cloudlet,

which has enough space for hosting at least one Avatar, for each iteration. The suboptimal cloudlet, denoted as $k^{'}$, of UE i is defined as the cloudlet that incurs the minimum weighted E2E delay among the cloudlets, which have enough resources to host at least one Avatar, i.e., $k^{'} = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$.

Denote Δc_i ($i \in \boldsymbol{\mathcal{I}}_1$) as the weighted E2E delay degradation by reallocating UE i's Avatar from its optimal cloudlet k into its suboptimal cloudlet $k^{'}$, i.e.:

$$\forall i \in \boldsymbol{\mathcal{I}}_1, \quad \Delta c_i = c_{ik} - c_{ik'}, \tag{26}$$

where $k = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_1 \right\}$ and $k^{'} = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$. Thus, the LEAD algorithm will select to reallocate a suitable UE i's Avatar, where $i = \arg\min_{\overline{i}} \left\{ \Delta c_{\overline{i}} | \overline{i} \in \boldsymbol{\mathcal{I}}_1 \right\}$, in each iteration in order to minimize the weighted E2E delay degradation. The iteration continues until $\boldsymbol{\mathcal{K}}_1 = \emptyset$. The detail of the LEAD algorithm is shown in Algorithm 2.

The complexity of each iteration in LEAD (i.e., from Step-5 to Step-9 in Algorithm 2) is $O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|) + O(|\boldsymbol{\mathcal{I}}|) + 3 \times O(1) = O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|)$, where $O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|)$ is the complexity of Step-5, $O(|\boldsymbol{\mathcal{I}}|)$ is the complexity of Step-6, and $3 \times O(1)$ is the complexity for executing Step-7 to Step-9. In the worst case scenario, the total number of iterations of LEAD is $|\boldsymbol{\mathcal{I}}|$. Thus, the complexity of LEAD is $O(|\boldsymbol{\mathcal{I}}|^2 \, |\boldsymbol{\mathcal{K}}|)$.

Note that the LEAD algorithm cannot guarantee that all the Avatars can be placed in one of its available cloudlets, i.e., it might happen that some Avatars, all of whose available cloudlets are full, cannot be hosted by one of its available cloudlets. Then, these Avatars will be placed in the central cloud (by default, the central cloud contains at least one replica of each UE's Avatar).

Lemma 4. *The LEAD algorithm terminates after a finite number of iterations, producing an feasible solution to the original Avatar handoff problem.*

Proof: Let $\xi = \sum_{k \in \boldsymbol{\mathcal{K}}_1} \left(\sum_{i \in \boldsymbol{\mathcal{I}}} z_{ik}^{'} - q_k \right)$. Assume $\boldsymbol{\mathcal{K}}_1 \neq \emptyset$ initially, and so $\xi > 0$. Then, in each iteration, the value of ξ is decreased by one because LEAD will reallocate UE i's Avatar from cloudlet k (i.e., set $z_{ik}^{'} = 0$), where $k = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_1 \right\}$, into cloudlet $k^{'}$, where $k^{'} = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$. Thus, ξ will be reduced to zero in a finite number of steps, and hence $\boldsymbol{\mathcal{K}}_1 = \emptyset$. ■

Similar to LEARN, the LEAD algorithm is also executed in a centralized manner to determine the locations of Avatars for their UEs in each time slot.

V. Simulation results

In order to evaluate our proposed replica placement algorithm and Avatar handoff algorithm, we have obtained data traces of more than 13,000 UEs and extracted their mobility in one day in Heilongjiang province in China[5]. The whole area contains 5,962 BSs and each UE's location is monitored during one day period. Specifically, each packet that is transmitted to/from a UE is monitored, and the packet analyzer extracts the BS information (i.e., the BS's ID and location) from each packet and considers the BS's location to be the current location of this UE (for instance, if a packet from the UE contains the information of BS-A, then we say the UE is currently associated with BS-A and the current location of the UE is BS-A's location). We apply these UE mobility traces to obtain the occurrence probability vector for each UE among BSs during the day (i.e., the values of $\boldsymbol{\mathcal{P}}_i = \{p_{ij} | j \in \boldsymbol{\mathcal{J}}\}$, where $p_{ij} = \frac{\textit{the total amount of time UE i is associated with BS j}}{\textit{one day period}}$). Meanwhile, we assume that each BS is attached to a cloudlet and the average E2E delay between a BS and a cloudlet (i.e., the value of t_{jk}, where $j \neq k$) is a function of the geographic distance between the BS and the cloudlet [33], i.e., $t_{jk} = 0.016 d_{jk} + 22.3$, where d_{jk} is the distance between BS j and cloudlet k in unit km and the unit of t_{jk} is in ms. Each BS's geographic location (i.e., the longitude and the latitude of the BS) is given by the UE mobility traces and each cloudlet is attached to a BS, and thus the value of d_{jk} is known; consequently, the value of t_{jk} can be calculated for all $j \in \boldsymbol{\mathcal{J}}$ and $k \in \boldsymbol{\mathcal{K}}$ based on the E2E delay model.

[5]The authors acknowledge the Center for Data Science of Beijing University of Posts and Telecommunications for providing these invaluable data traces.

Algorithm 2 LEAD algorithm

Input: 1) The replica placement vector for each UE, i.e., $\boldsymbol{\mathcal{X}}_i^{opt} = \left\{ x_{ik}^{opt} | k \in \boldsymbol{\mathcal{K}} \right\}, \forall i \in \boldsymbol{\mathcal{I}}$. 2) The average E2E delay vector, i.e., $\boldsymbol{\mathcal{T}} = \{t_{jk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\}$. 3) The location indicator vector for all UEs in the current time slot, i.e., $\boldsymbol{\mathcal{L}} = \{l_{ij} | i \in \boldsymbol{\mathcal{I}}, j \in \boldsymbol{\mathcal{J}}\}$.

Output: Avatar location indicator vector for all UEs, i.e., $\boldsymbol{\mathcal{Z}}^{'} = \left\{ z_{ik}^{'} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}} \right\}$.

```
1:  Initialize $z_{ik}^{'}$ by solving the relaxed problem (i.e., $\boldsymbol{P4}$) based on Eq. (25).
2:  Initialize the cloudlet sets $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ based on their definition.
3:  Initialize the UE set $\boldsymbol{\mathcal{I}}_1$ based on its definition.
4:  while $\boldsymbol{\mathcal{K}}_1 \neq \emptyset$ do
5:      $\forall i \in \boldsymbol{\mathcal{I}}_1$, calculate $\Delta c_i$ based on Eq. (26);
6:      Find UE $i$, where $i = \arg\min_{\overline{i}} \{\Delta c_{\overline{i}} | \overline{i} \in \boldsymbol{\mathcal{I}}_1\}$;
7:      Reallocate UE $i$'s Avatar from its optimal cloudlet $k$ (i.e., set $z_{ik}^{'} = 0$) into its suboptimal cloudlet $k^{'}$, (i.e., set $z_{ik'}^{'} = 1$);
8:      Update the cloudlet sets $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$;
9:      Update the UE set $\boldsymbol{\mathcal{I}}_1$;
10: end while
11: return $\boldsymbol{\mathcal{Z}}^{'}$.
```

A. Performance of the LEARN algorithm

First, we simulate the proposed LEARN algorithm based on the mentioned UE mobility trace. Specifically, we first calculate the value of p_{ij} based on the mentioned UE mobility trace. Then, by taking $\boldsymbol{\mathcal{P}}_i$ and $\boldsymbol{\mathcal{T}}$ as input parameters, we further obtain the replica placement vector for each UE by applying the LEARN algorithm. Consequently, the average

E2E delay for all the UEs during the day is derived given the replica placement vector for each UE. For comparisons, we considered the scenario that all the UEs' Avatars are located in the central data center (i.e., UEs' Avatars cannot migrate when UEs are roamed among BSs), which is placed in the southeast point of the area. In this scenario, we also calculate the average E2E delay between UEs and the data center during the day.

As shown in Fig. 4, given the number of replicas (i.e., the value of κ), the average E2E delay achieved by LEARN is significantly reduced as compared to that of the traditional big data network (in which the UEs access their Avatars in the remote cloud/data center via the Internet). Moreover, as the number of replicas increases, the average E2E delay is reduced accordingly. Specifically, as compared to the traditional data center network, the average E2E delay achieved by LEARN is improved by 75.79%, 93.12%, 97.27%, and 98.59% when the value of is selected to be 1, 2, 3, and 4, respectively. Note that $\kappa = 1$ (i.e., there is one replica for each UE) indicates the location of each Avatar is fixed (i.e., the Avatar is placed in the location where its UE most visit) and the Avatar cannot handoff among the cloudlets because no extra replicas are placed in other cloudlets. Also, when the value of κ is greater than 4, the decrement of the E2E delay by increasing the value of κ is not significant. We further analyze the mobility trace by calculating at least how many locations, i.e., the BSs' coverage area, that each UE will stay over 90%, 95% and 99% of the time during the day, respectively. As shown in Fig. 5, 92.22%, 86.93% and 75.65% of the UEs spend 90%, 95% and 99% of the time during the day (in terms of 21.6, 22.8 and 23.76 hours) to stay at only four locations, respectively. Thus, placing five replicas for the UEs may not significantly benefit the average E2E delay during the day as compared to placing four replicas. Note that placing more replicas for each Avatar may increase the CAPEX to the cloudlet network provider by deploying more storage resources. Also, allocating more replicas for each Avatar may generate more synchronous traffic, and thus increase the traffic load of the cloudlet network.

B. *Performance of the LEAD algorithm*

We further evaluate the performance of our proposed LEAD algorithm. Each Avatar's replicas have already been placed to the corresponding cloudlets, which are calculated by the LEARN algorithm. Still, UEs' mobilities and the locations of the BSs follow the mobility traces that we have sampled from the real world. By applying the mobility traces, the location indicator vector for all UEs (i.e., the values of $\mathcal{L}$) in different time slots during the day can be obtained. Initially, we setup the capacity of all the cloudlet to be the same, i.e., $\forall k \in \mathcal{K}, q_k = 10$ (each cloudlet can host at most 10 Avatars in each time slot). We first test the average E2E delay between UEs and their Avatars during the day. As shown in Fig. 7, as compared to the results of the LEARN algorithm[6] (which generates the optimal average E2E delay between UEs and their Avatars without considering the capacity constraints), the average E2E delay generated by the LEAD algorithm is increased because some Avatars cannot handoff to their optimal cloudlets (due to the cloudlet capacity limitation) when the UEs roam away. Consequently, these Avatars need to handoff to their suboptimal cloudlets or to the remote data center. Note that as the number of replicas increases, the average E2E delay decreases accordingly. This is because as the number of replicas increases, the Avatars, whose optimal cloudlets exceed their capacity limitation, have higher probability to handoff to their suboptimal cloudlets with lower E2E delay. For instance, as shown in Fig. 6, assume there are two Avatar's replicas that have been optimally placed in cloudlet A and cloudlet B. Suppose the optimal location of the Avatar is cloudlet A at the current time slot but the cloudlet A is full, and so the Avatar needs to handoff to the suboptimal cloudlet, which is cloudlet B; this may increase the E2E delay between the Avatar and its UE, and we denote the E2E delay increment as t_{A-B}. Then, if there are one more Avatar's replicas that have been placed in the cloudlet C, and so the Avatar can be handed off to the suboptimal cloudlet, which can be cloudlet B or cloudlet C. Thus, the E2E delay increment by handing off the Avatar to the suboptimal cloudlet is $\min\{t_{A-B}, t_{A-C}\}$, where t_{A-C} is the E2E delay increment by handing off the Avatar to cloudlet C. Obviously, $t_{A-B} \geq \min\{t_{A-B}, t_{A-C}\}$. Therefore, the average E2E delay between UEs and their Avatars decreases when the number of the replicas increases.

Note that, as mentioned in Sec. IV-B, the LEAD algorithm cannot guarantee that all the Avatars can be handed off to their available cloudlets, i.e., some Avatars need to be handed off to the remote data center, which is the main factor of increasing the average E2E delay because the average E2E between the UEs and the remote data center reaches 254.8 ms, which is significantly longer than the average E2E delay produced by the LEAD algorithm. Thus, we further test the average number of the *remote Avatars*, which are defined as the Avatars that have to be handed off to the remote data center, during the day. As shown in Fig. 8, the average number of the *remote Avatars* decreases as the number of each Avatar's replicas increases. This is because that the probability (that all the Avatar's available cloudlets are full) decreases as the number of the Avatar's replicas increases.

As mentioned previously, the average E2E delay of the LEAD algorithm is longer than that of the LEARN algorithm because LEARN does not consider the capacity limitation of the cloudlets. In other words, owing to the capacity constraints, some Avatars cannot be handed off to their optimal cloudlets, thus resulting in the average E2E delay growth in LEAD. In order to study how the cloudlet capacity impacts the performance of the LEAD algorithm. We further record the average E2E delay of the LEAD algorithm by changing the capacity of each cloudlet. As shown in Fig 9, when the cloudlet capacity increases (from 6 to 16), the average E2E delay of the LEAD algorithm improves tremendously. As the cloudlet capacity reaches 16, the average E2E delay of the LEAD algorithm does not improve significantly. This is because most of the Avatars can be handed off to their optimal cloudlets as their UEs roam away. Although increasing the cloudlet

[6]We consider the average E2E delay of the LEARN algorithm as the lower bound in terms of the best case scenario of the LEAD algorithm.

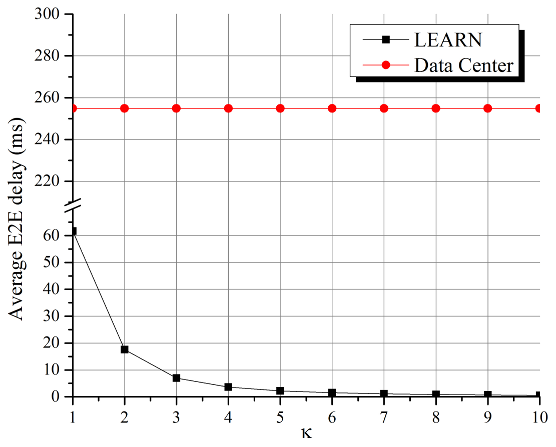


Fig. 4. The average E2E delay of the cloudlet network by applying LEARN and that of the traditional big data network.

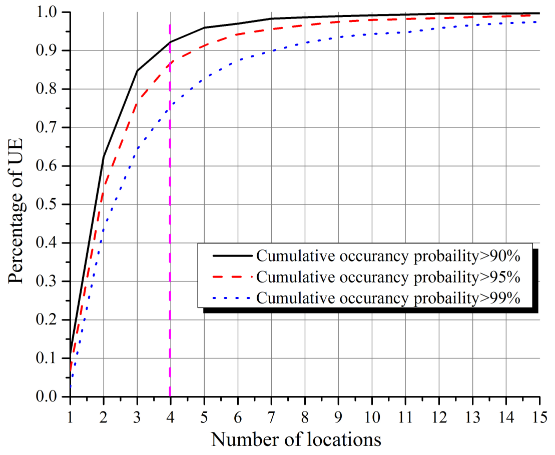


Fig. 5. The statistical results of the UE mobility trace.

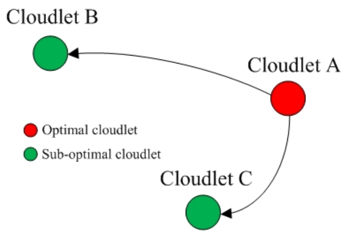


Fig. 6. The illustration of the average E2E delay reduction.

capacity can improve the average E2E delay, the average cloudlet utilization[7] is reduced accordingly. As shown in Fig. 10, the average cloudlet utilization drops from 35% to 10.9% as the cloudlet capacity increases from 6 to 26.

Therefore, there is a tradeoff between the average E2E delay and the cloudlet utilization. In order to optimize the tradeoff, the capacity of different cloudlets should be varied. Specifically, the cloudlets, whose connected BSs have higher UE density (such as shopping malls and public transportations), should have larger capacity, and vice versa. Thus, in the future, we will try to design a cloudlet deployment strategy to determine the capacity of each cloudlet such that the average cloudlet utilization can be maximized and the average E2E delay is guaranteed.

VI. Related works

Recently, telecommunications vendors have shown the great interest on the concept of mobile edge computing (MEC). European Telecommunications Standards Institute (ETSI) created an industry initiative on MEC to standardize the MEC platform by utilizing the concept of cloudlet. Also, in the academic area, many works [35]–[39] have proposed to utilize the cloudlet to reduce the E2E delay between users and computing resources, and thus improve the performance of MCC applications as well as big data networking. Chen *et al.* [40] implemented a cognitive assistance application (which provides step-by-step visual guidance for users to implement a complex task) running in a wearable device (such as Google Glass) with the help of cloudlet processing. The cloudlet is deployed one wireless hop away to guarantee the stringent E2E delay required by the applications. Quwaider and Jararweh [41] developed a cloudlet-based wireless body area network. The data streams generated by the users are transmitted to the nearest cloudlet through WiFi. The cloudlet stores and processes the data streams locally to reduce the latency as well as the communications power consumption as compared to the traditional cloud-based wireless body area network. Xu *et al.* [42] proposed an efficient algorithm to optimally place a fixed number of cloudlets among the wireless access points in the wireless metropolitan area network while minimizing the average access delay between mobile users and the cloudlets. Ceselli *et al.* [43] also tried to optimally deploy a number of cloudlets among the aggregation nodes and core nodes in the cellular network in order to minimize the capital cost (i.e., installation costs) of the cloudlet providers while considering the latency between users and their VMs in the cloudlets.

All the above papers do not consider how to optimally handoff users' Avatars/VMs among cloudlets when users are roaming in the network. Dynamical Avatar/VM handoff is critical in the cloudlet network because the E2E delay between the users and their Avatars/VMs may become worse when users roam away. Our previous work [15], [44] presented a green cloudlet network architecture in which all the cloudlets are powered by both green energy and on-grid energy. In order to minimize the on-grid energy consumption, we proposed to migrate the Avatars to the cloudlets with more green energy generation and less energy demands while guaranteeing the E2E delay requirement between users and their Avatars. However, this work did not take the virtual disk migration into consideration during the Avatar migration process and the virtual disk migration may incur long E2E delay as well as the network congestion as explained in Sec. II. In order to avoid the virtual disk migration during the handoff process, Ha *et al.* [21] proposed that a number of base VM images, which contains a set of widely used software/content, are pre-

[7] $cloudlet\,utilization = \frac{total\ number\ of\ Avatars\ hosted\ by\ the\ cloudlet}{the\ cloudlet\ capacity}$

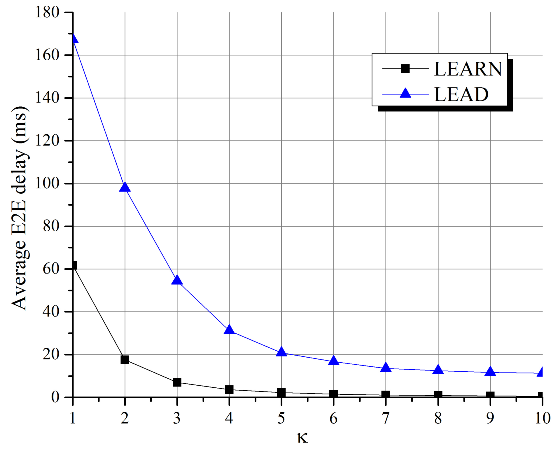


Fig. 7. The average E2E delay of LEAD and LEARN.

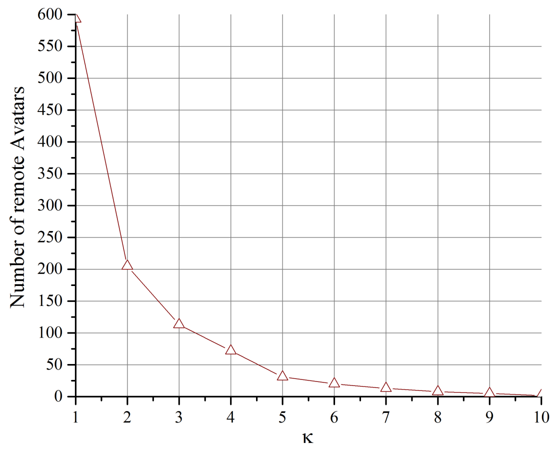


Fig. 8. The average number of *remote Avatars*.

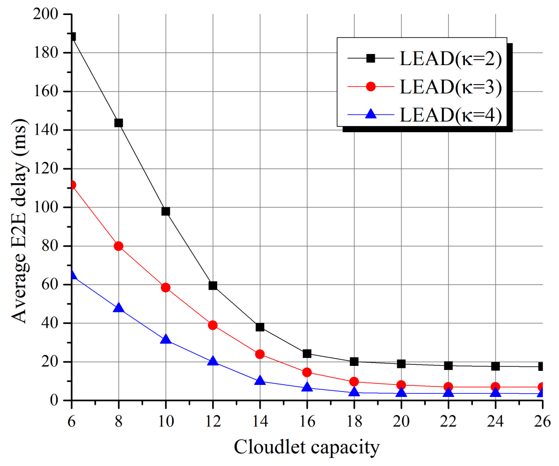


Fig. 9. The average E2E delay over different cloudlet capacities.

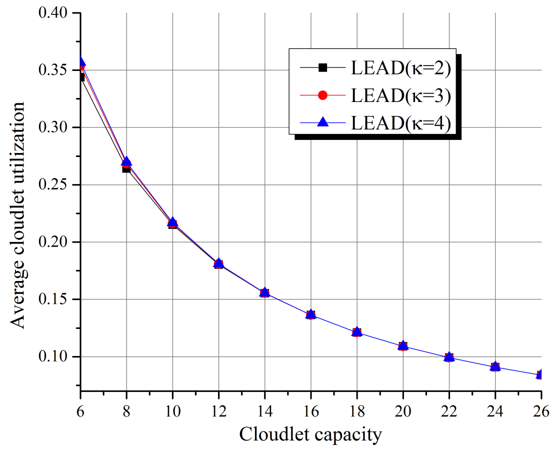


Fig. 10. The average cloudlet utilization over different cloudlet capacities.

deployed in each cloudlet. Thus, once the user's VM is handed off to another cloudlet, only the VM overlay (which is defined as the difference between the user's VM and the base VM images in the cloudlet) need to be migrated over the network. However, the VM handoff time is still rather long if the user's VM has a huge amount of private VM overlay. Moreover, in our cloudlet network, each Avatar should store and process its UE's private data (such as photos, GPS information as well as sensing data streams), which would incur larger VM overlay.

As compared to these previous efforts, this paper presents several enhancements. First, we have proposed the cloudlet network architecture by bringing the computing and storage resources close to UE in order to reduce the E2E delay between UEs and computing resources. Second, each UE is assigned a dedicated VM in terms of the Avatar to process its own data and applications. Meanwhile, each UE can access its Avatar seamlessly based on the cloudlet network. Third, in order to facilitate Avatar handoff (by avoiding the virtual disk migration) and maintain low average E2E delay between UEs and their Avatars, we have designed the LEARN algorithm to place a number of replicas into suitable cloudlets. Fourth, considering the capacity limitation of each cloudlet, we have proposed the LEAD algorithm to optimize the locations of all the Avatars in each time slot in order to minimize the average E2E delay for all the UEs and their Avatars during each time slot.

VII. Conclusion

In this paper, we have proposed the cloudlet network architecture to facilitate big data networking as well as mobile cloud computing. Specifically, each UE can access its own Avatar, which is considered as private computing and storage resources for the UE, with the low E2E delay. In order to maintain the low E2E delay when UEs roam away, their Avatars should be handed off among cloudlets accordingly. However, migrating the high volume of the Avatar's virtual disk during the handoff process incurs unbearable handoff time, which may significantly degrade the E2E delay as well as the performance of the Avatar. Also, migrating the Avatar's virtual disk during the handoff process may tremendously increase the traffic in the SDN based cellular core. Thus, in order to avoid the virtual disk migration during the handoff process, we have proposed to place a number of replicas of the Avatar's virtual disk among the cloudlets so that the Avatar can be handed off among its available cloudlets (which contain one of the Avatar's replicas) based on its UE's location. We have designed the LEARN algorithm to optimally place the replicas among the cloudlets for each Avatar so that the average E2E delay between the Avatar and its UE is minimized during the day. Moreover, after optimally deploying the replicas for each Avatar, we have designed the LEAD algorithm to determine the locations of all the Avatars in each time slot so that the average E2E delay between all the UEs and their Avatars is minimized in each time slot, while the capacity of the each cloudlet is not violated. The simulation results demonstrate that applying the LEARN algorithm in the cloudlet network architecture can significantly reduce the average E2E delay between UEs and their Avatars during the day as compared to the traditional centralized cloud architecture (i.e., all the UEs' Avatars are located in the central cloud). Furthermore, the LEAD algorithm can still maintain the low average E2E delay by selecting suitable parameters in terms of the number of the replicas for each Avatar and the capacity of each cloudlet. Note that handing off a UE's Avatar among its available cloudlets when the UE roams away may still generate an extra overhead [10], e.g., the extra traffic for migrating Avatar's memory to the destination cloudlet. Thus, there is a tradeoff between minimizing the E2E delay (between a UE and its Avatar) and minimizing the extra overhead. To optimize this tradeoff, various aspects shall be considered, i.e., the memory utilization of the UE's Avatar, the network condition in the SDN-based cellular core network, etc.

We have demonstrated that the tradeoff between the average E2E delay and the average cloudlet utilization through the simulations. Thus, in the future, we will study the cloudlet placement problem, i.e., determining the suitable capacity for each cloudlet, to optimize this tradeoff. Moreover, the number of the replicas can also be varied among different UEs and the suitable number of replicas can be selected based on the UE's behavior.

References

[1] Q. Han, S. Liang, and H. Zhang, "Mobile Cloud Sensing, Big Data, and 5G Networks Make an Intelligent and Smart World," in *IEEE Network*, vol. 29, no. 2, pp. 40–45, Mar.–Apr. 2015.

[2] M. Musolesi, "Big Mobile Data Mining: Good or Evil?," in *IEEE Internet Computing*, vol. 18, no. 1, pp. 78–81, Jan.–Feb. 2014.

[3] X. Yi, F. Liu, J. Liu, and H. Jin, "Building a Network Highway for Big Data: Architecture and Challenges," in *IEEE Network*, vol. 28, no. 4, pp. 5–13, July–August 2014.

[4] J. Liu, F. Liu, and N. Ansari, "Monitoring and Analyzing Big Traffic Data of a Large Scale Cellular Network with Hadoop," in *IEEE Network*, vol. 28, no. 4, pp. 32–39, July–August 2014.

[5] E. Baccarelli, *et al.*, "Energy-efficient Dynamic Traffic Offloading and Reconfiguration of Networked Data Centers for Big Data Stream Mobile Computing: Review, Challenges, and a Case Study," in *IEEE Network*, vol. 30, no. 2, pp. 54–61, March–April 2016.

[6] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] M. Isard, *et al.*, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 59–72, 2007.

[8] Apache storm. [Online]. Available: https://storm.apache.org/, date of access: May 1, 2017.

[9] X. Sun, N. Ansari, and R. Wang, "Optimizing Resource Utilization of a Data Center," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2822-2846, Fourthquarter 2016.

[10] X. Sun and N. Ansari,"PRIMAL: PRofIt Maximization Avatar pLacement for Mobile Edge Computing," in *Proceedings of IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 23–27, 2016, pp. 1–6.

[11] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and Flexible Cellular Core Network Architecture," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, Santa Barbara, CA, Dec. 09–12, 2013, pp. 163–174.

[12] X. Sun and N. Ansari, "EdgeIoT: Mobile Edge Computing for the Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, December 2016.

[13] X. Sun and N. Ansari, "Latency Aware Workload Offloading in the Cloudlet Network," in *IEEE Communications Letters*, doi: 10.1109/LCOMM.2017.2690678, *early access*.

[14] Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) Specification. [Online]. Available: http://www.3gpp.org/specifications-groups/ran-plenary/ran2-radio-layer-2-and-radio-layer-3-rr, date of access: May 1, 2017.

[15] X. Sun, N. Ansari, and Q. Fan, "Green Energy Aware Avatar Migration Strategy in Green Cloudlet Networks," in *7th IEEE International Conference on Cloud Computing Technology and Science (Cloudcom))*, Vancouver, Canada, Nov. 30–Dec. 3, 2015, pp. 139–146.

[16] C. Borcea, *et al.*, "Avatar: Mobile Distributed Computing in the Cloud," in *3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud'15)*, San Francisco, CA, Mar. 30-Apr. 3, 2015, pp. 151–157.

[17] B. G. Chun, *et al.*, "Clonecloud: Elastic Execution between Mobile Device and Cloud," in *Proceedings of the sixth conference on Computer systems*, Salzburg, Austria, Apr. 10–13, 2011, pp. 301–314.

[18] X. Sun and N. Ansari, "Energy-Optimized Bandwidth Allocation Strategy for Mobile Cloud Computing in LTE Networks," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, 2015, pp. 2120–2125.

[19] E. Cuervo, *et al.*, "MAUI: Making Smartphones Last Longer with Code Offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, San Francisco, CA, Jun. 15–18, 2010, pp. 49–62.

[20] S. R. Ellis, K. Mania, B. D. Adelstein, and M. I. Hill, "Generalizeability of Latency Detection in a Variety of Virtual Environments," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, New Orleans, LA, Sep. 20–24, 2004, vol. 48, no. 23, pp. 2632–2636.

[21] K. Ha, *et al.*, "Adaptive VM Handoff Across Cloudlets," Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.

[22] W. Zhang, K. T. Lam and C. L. Wang, "Adaptive Live VM Migration over a WAN: Modeling and Implementation," in *2014 IEEE 7th International Conference on Cloud Computing*, Anchorage, AK, 2014, pp. 368–375.

[23] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and Mobility: User Movement in Location-based Social Networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, San Diego, CA, Aug. 21–24, 2011, pp. 1082–1090.

[24] W. Su, S. J. Lee, and M. Gerla. "Mobility Prediction in Wireless Networks," in *21st Century Military Communications Conference Proceedings(MILCOM)*, Los Angeles, CA, Oct. 22–25, 2000, pp. 491–495.

[25] A. Nadembega, A. Hafid, and T. Taleb, "A Destination and Mobility Path Prediction Scheme for Mobile Networks," in *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2577–2590, June 2015.

[26] N. L. M. Adrichem, C. Doerr, and F. Kuipers, "Opennetmon: Network Monitoring in OpenFlow Software-Defined Networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May. 05–09, pp. 1–8.

[27] C. Yu, *et al.*, "Software-Defined Latency Monitoring in Data Center Networks," in *2015 16th International Conference on Passive and Active Measurement (PAM)*, New York, NY, Mar. 19–20, 2015, pp. 360-372.

[28] Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, *Location Science*, Springer, ISBN: 978-3-319-13111-5, 2015.

[29] O. Kariv and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems II: The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 539–560, 1979.

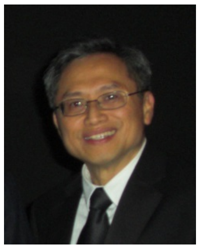
[30] Mark S. Daskin, '*Network and Discrete Location: Models, Algorithms, and Applications*, John Wiley & Sons, ISBN: 978-0-470-90536-4, 2011.

[31] M. Held, P. Wolfe, and H. P. Crowder, "Validation of Subgradient Optimization," *Mathematical programming*, vol. 6, no. 1, pp. 62–88, 1974.

[32] M. Held and R. M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical programming*, vol. 1, no. 1, pp. 6–25, 1971.

[33] R. Landa, *et al*., "The Large-Scale Geography of Internet Round Trip Times," *IFIP Networking Conference*, Brooklyn, NY, May 22–24, 2013, pp. 1-9.
[34] S. Sahni and T. Gonzalez, "P-Complete Approximation Problems," in *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 555-565, 1976.
[35] G. A. Lewis, *et al*., "Cloudlet-based Cyber-Foraging for Mobile Systems in Resource-Constrained Edge Environments," in *Companion Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, Indian, May 31–Jun. 07, 2014, pp. 412–415.
[36] Z. Xu, *et al*., "Efficient Algorithms for Capacitated Cloudlet Placements," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, Oct. 1 2016.
[37] Y. Zhang, D. Niyato, and P. Wang, "Offloading in Mobile Cloudlet Systems with Intermittent Connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529.
[38] T. G. Rodrigues, *et al.*, "Towards a Low-Delay Edge Cloud Computing through a Combined Communication and Computation Approach," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Montreal, QC, Canada, 2016, pp. 1–5.
[39] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing through VM Migration and Transmission Power Control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, May 1 2017.
[40] Z. Chen, *et al*., "Early Implementation Experience with Wearable Cognitive Assistance Applications," in *Proceedings of the 1st ACM Workshop on Wearable Systems and Applications (WearSys)*, Florence, Italy, May 19–22, 2015, pp. 33–38.
[41] M. Quwaider and Y. Jararweh, "Cloudlet-based Efficient Data Collection in Wireless Body Area Networks," *Simulation Modelling Practice and Theory*, vol. 50, pp. 57–71, 2015.
[42] Z. Xu, W. Liang, W. Xu, M. Jia and S. Guo, "Capacitated Cloudlet Placements in Wireless Metropolitan Area Networks," *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, Oct. 26–29, 2015, pp. 570–578.
[43] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet Network Design Optimization," in *IFIP Networking Conference (IFIP Networking)*, Toulouse, France, May 20–22, 2015, pp. 1–9.
[44] X. Sun and N. Ansari, "Green Cloudlet Network: A Distributed Green Mobile Cloud Network," *IEEE Network*, vol. 31, no. 1, pp. 64–70, January/February 2017.

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 edge computing and communications, Internet of Things, and cloud computing.



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.

Professor Ansari has 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.

Professor Ansari 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 over 30 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.