Latency Aware Workload Offloading in the Cloudlet Network

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

***Abstract*—We propose a novel network architecture by leveraging the cloudlet concept, the Software Defined Networking (SDN) technology, and the cellular network infrastructure to bring the computing resource to the mobile edge. In order to minimize the average response time for Mobile Users (MUs) in offloading their application workloads to the geographically distributed cloudlets, we propose the Latency awarE workloAd offloaDing (LEAD) strategy to allocate MUs' application workloads into suitable cloudlets. Simulation results demonstrate that LEAD incurs the lowest average response time as compared to two other existing strategies.**

***Index Terms*—Cloudlet network, application workload offloading, delay, software defined networking**

I. Introduction

MOBILE cloud computing is a promising paradigm to meet the growing demand for computing intensive applications and services from Mobile Users (MUs). However, offloading the Mobile Cloud Computing (MCC) applications' workloads from MUs to the remote cloud may significantly degrade the Quality-of-Experience in terms of the response time of MCC applications because the network delay between MUs and the remote cloud may be too long to satisfy the stringent requirement of MCC applications. In order to reduce the application's response time, a new framework, referred to as the *cloudlet network*, has been proposed to bring the computing resources from the remote cloud to the mobile edge [1], [2]. As shown in Fig. 1, each Base Station (BS) is connected to a cloudlet, which consists of a number of interconnected physical machines. Each cloudlet provides powerful computing resources to its local MUs. The placement of cloudlets is flexible, i.e., a cloudlet can directly connect to a BS via high speed fibers, or to an edge switch so that multiple BSs can share the computing resources in the same cloudlet. The Software Defined Networking (SDN) based cellular core is introduced to establish an efficient routing path between two BSs. The proposed architecture facilitates the MCC application workload offloading process by enabling MUs to access the computing resources with the low network delay.

One of the main objectives of the cloudlet network is to improve the application response time in terms of the delay for offloading the MCC application workloads to the computing resources. The response time includes the network delay by uploading the application workloads from an MU to a cloudlet and downloading the results from the cloudlet to the MU, the queueing delay, and processing delay for executing the application workloads in the cloudlet. Note that the cloudlet delay reflects the queueing delay and processing delay in the rest of the paper. MUs can associate with the nearby cloudlet to offload their application workloads in order to minimize the network delay; however, MUs are roaming among different BSs' coverage areas, and so it is beneficial to adjust different cloudlets to serve MUs in order to reduce the network delay. On the other hand, the computing resource of each cloudlet is limited, and thus the cloudlet may not have enough resources to handle the application workloads from its local MUs; this can significantly increase the cloudlet delay for the cloudlet to handle application requests, and thus increase the application response time. Therefore, it is necessary to re-assign the application workloads to another cloudlet if the original cloudlet is overloaded.

X. Sun and N. Ansari are with Advanced Networking Lab., Department of Electrical & Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA. E-mail:{xs47, nirwan.ansari}@njit.edu.

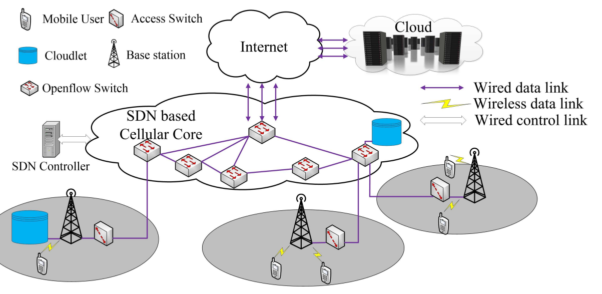

Fig. 1. The cloudlet network architecture.

Current research focuses on how to offload the application workloads from MUs to cloudlets. Some try to design an offloading decision mechanism to determine whether to locally execute the application or offload the application workloads of an MU to a nearby cloudlet or the remote cloud in order to minimize the energy consumption of the MU and/or minimize the latency [3], [4]. Sun and Ansari [5] proposed the computing resources assignment algorithm in the cloudlet network to minimize the network delay, but they did not consider the cloudlet delay. Jia *et al.* [6] designed an application workload assignment algorithm to balance the workloads among geo-distributed cloudlets, but they did not consider the network delay. Elgazzar *et al.* [7] proposed the Follow-Me-Provider selection scheme to choose a suitable resource provider (i.e., a local mobile device, a nearby mobile cloud, a cloudlet, or the remote cloud) for executing an application task based on the context information (such as computational capacity of the resource provider, the available bandwidth for the communications, etc.); however, they did not consider the network delay either. To our best knowledge, how to optimally select a cloudlet in the cloudlet network to handle an MU's application workloads in order to minimize the average response time (i.e., by both considering the network delay and the cloudlet delay) for all MUs remains a challenging problem.

In this letter, we introduce a Latency awarE workloAd offloaDing (LEAD) strategy in the context of the proposed cloudlet network architecture to minimize the average application response time among MUs. The main contributions are listed as follows: 1. We propose a novel cloudlet network architecture to enable MUs to offload their application workloads to nearby cloudlets. 2. We formulate the problem of minimizing the average response time by offloading MUs' application workloads to suitable cloudlets and prove the problem to be NP-hard. 3. We design the novel LEAD strategy to solve the problem and demonstrate the performance of LEAD via simulations.

II. Average response Time

In order to estimate the average response time for each MU, we will analyze the average network delay and the average cloudlet delay in this section.

A. Average network delay

The network delay of offloading an MU's application request to a cloudlet, denoted as T^{net}, comprises: 1) $T^{trans}_{MU\to BS}$: the transmission delay for uploading the application request from the MU to its associated BS, 2) $T^{net}_{BS\to cloudlet}$: the network delay for transmitting the request from the MU's BS to the MU's cloudlet (which is assigned to serve the MU)[1], 3) $T^{net}_{cloudlet\to BS}$: the network delay for transmitting the results from the MU's cloudlet to the MU's BS, and 4) $T^{trans}_{BS\to MU}$: the transmission delay for downloading the results from the MU's BS to the MU. Thus, $T^{net} = T^{trans}_{MU\to BS} + T^{trans}_{BS\to MU} + T^{RTT}_{BS\leftrightarrow cloudlet}$, where $T^{RTT}_{BS\leftrightarrow cloudlet}$ is the Round Trip Time (RTT) between the BS and the cloudlet, i.e., $T^{RTT}_{BS\leftrightarrow cloudlet} = T^{net}_{BS\to cloudlet} + T^{net}_{cloudlet\to BS}$. Note that no matter which cloudlet serves the MU, it will not affect the values of $T^{trans}_{MU\to BS} + T^{trans}_{BS\to MU}$. Thus, we will not consider the transmission delay between the MU and its associated BS in the rest of the paper, i.e., $T^{net} = T^{RTT}_{BS\leftrightarrow cloudlet}$.

Denote $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{J}}$, and $\boldsymbol{\mathcal{K}}$ as the set of MUs, BSs, and cloudlets, respectively. Denote x_{ik} as a binary variable to indicate whether the application workloads generated by MU i are handled by cloudlet k ($x_{ik} = 1$) or not ($x_{ik} = 0$). Denote t_{jk} as the RTT between BS j and cloudlet k. Note that the value of t_{jk} ($j \neq k$) can be measured and recorded by the SDN controller periodically [8]. Denote y_{ij} as a binary indicator to imply MU i being in BS j's coverage area ($y_{ij} = 1$) or not ($y_{ij} = 0$). Then, the average network delay of MU i is

$$T^{net}_i = \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} y_{ij}t_{jk}x_{ik}. \tag{1}$$

B. Average cloudlet delay

As the MU's application requests arrive in the MU's cloudlet, the cloudlet would assign the amount of computing resources to process the MU's application requests. Thus, we model the processing of application requests from MUs by each cloudlet as a queueing model and assume the application request generations for MU i (where $i \in \boldsymbol{\mathcal{I}}$) follow a Poisson distribution with the average application generation rate equal to λ_i [9]. Therefore, the application request arrivals to cloudlet k (where $k \in \boldsymbol{\mathcal{K}}$), which is the sum of the application request generations of the MUs associated with cloudlet k, also follows a Poisson distribution with the average application arrival rate equal to $\sum\limits_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik}$. Meanwhile, we assume the service time of cloudlet k (where $k \in \boldsymbol{\mathcal{K}}$) for executing application requests from MUs is exponentially distributed with the average service time equal to $1/u_k$, where u_k is the average service rate of cloudlet k. Note that we consider a cloudlet as one entity to handle the application requests from MUs. Although a cloudlet may comprise a number of interconnected Physical Machines (PMs) to process the incoming application requests, we are focusing on the coarse-grained workload offloading scheme in this letter, i.e., we try to allocate the workloads among cloudlets. By considering a cloudlet as an entity, it is therefore appropriate to model the processing of application requests from MUs by a cloudlet as an M/M/1-PS queueing model. Consequently, we can derive the average cloudlet delay of MU i by offloading its application requests to cloudlet k as

$$T^{cloudlet}_{ik} = \frac{1}{u_k - \sum\limits_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}. \tag{2}$$

Thus, the average cloudlet delay of MU i is

$$T^{cloudlet}_i = \sum_{k\in\boldsymbol{\mathcal{K}}} T^{cloudlet}_{ik}x_{ik} = \sum_{k\in\boldsymbol{\mathcal{K}}}\frac{x_{ik}}{u_k - \sum\limits_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}. \tag{3}$$

Based on Eq. 1 and Eq. 3, the average response time of MU i, denoted as τ_i, is

$$\tau_i = \sum_{k\in\boldsymbol{\mathcal{K}}}\left(\sum_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk} + \frac{1}{u_k - \sum\limits_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}\right)x_{ik}. \tag{4}$$

III. Problem formulation

The main purpose of the cloudlet network is to minimize the average response time for all the MUs in the network. Thus, we formulate the problem, i.e., minimizing the average response time by offloading MUs' application requests to suitable cloudlets, as follows:

$$\boldsymbol{P0}: \underset{x_{ik}}{\operatorname{argmin}} \frac{1}{|\boldsymbol{\mathcal{I}}|}\sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{k\in\boldsymbol{\mathcal{K}}}\left(\sum_{j\in\boldsymbol{\mathcal{J}}} y_{ij}t_{jk} + \frac{1}{u_k - \sum\limits_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}\right)x_{ik} \tag{5}$$

$$s.t.\quad \forall k\in\boldsymbol{\mathcal{K}},\quad u_k - \sum_{i\in\boldsymbol{\mathcal{I}}}\lambda_i x_{ik} > 0, \tag{6}$$

$$\forall i\in\boldsymbol{\mathcal{I}},\quad \sum_{k\in\boldsymbol{\mathcal{K}}} x_{ik} = 1, \tag{7}$$

$$\forall i\in\boldsymbol{\mathcal{I}}, \forall k\in\boldsymbol{\mathcal{K}},\quad x_{ik}\in\{0,1\}, \tag{8}$$

where the objective is to minimize the average response time among MUs in the network. Constraint (6) is to guarantee the

[1]We assume that each MU will be allocated the amount of resources in only one cloudlet in a time slot. Allocating the resources in different cloudlets during the same time slot will introduce extra overheads for the MU.

average service rate to be less than the average arrival rate for each cloudlet in order to make the system stable. Constraint (7) is to ensure that every MU is only served by one cloudlet.

Theorem 1. *The problem of minimizing the average response time by offloading MUs' application requests to suitable cloudlets (i.e., $\boldsymbol{P0}$) is NP-hard.*

Proof: In order to prove that $\boldsymbol{P0}$ is an NP-hard problem, we can demonstrate that its decision problem is NP-complete. The decision problem of $\boldsymbol{P0}$ can be described as follows: given a positive value b, is it possible to find a feasible solution $\boldsymbol{\mathcal{X}} = \{x_{ik} | \forall i \in \boldsymbol{\mathcal{I}}, \forall k \in \boldsymbol{\mathcal{K}}\}$ such that $\frac{1}{|\boldsymbol{\mathcal{I}}|}\sum\limits_{i \in \boldsymbol{\mathcal{I}}}\sum\limits_{k \in \boldsymbol{\mathcal{K}}}\left(\sum\limits_{j \in \boldsymbol{\mathcal{J}}} y_{ij}t_{jk} + \frac{1}{u_k - \sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}\right)x_{ik} \leq b$ and Constraint (6), (7), and (8) are satisfied as well?

In order to demonstrate the decision problem of $\boldsymbol{P0}$ is NP-complete, only two cloudlets are considered in the network and the average service rate of the two cloudlets are the same, i.e., $u_1 = u_2 = \frac{1}{2}\sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i + \varepsilon$, where ε is a very small value (i.e., $\varepsilon \ll \frac{1}{2}\lambda^{\min}$, where $\lambda^{min} = \min\{\lambda_i | i \in \boldsymbol{\mathcal{I}}\}$). Meanwhile, assume that $b \to +\infty$, i.e., $\frac{1}{|\boldsymbol{\mathcal{I}}|}\sum\limits_{i \in \boldsymbol{\mathcal{I}}}\sum\limits_{k \in \boldsymbol{\mathcal{K}}}\left(\sum\limits_{j \in \boldsymbol{\mathcal{J}}} y_{ij}t_{jk} + \frac{1}{u_k - \sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{ik}}\right)x_{ik} \leq b$ is always satisfied for any solution of $\boldsymbol{\mathcal{X}}$. Then, the decision problem can be transformed into a partition problem. That is whether all the MUs can be partitioned into two sets such that the average application request arrival rates of the two sets are the same, i.e., $\sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{i1} = \sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{i2} = \frac{1}{2}\sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i$. Therefore, the partition problem is reducible to the decision problem of $\boldsymbol{P0}$. Note that the partition problem is a well-known NP-complete problem. Consequently, the decision problem of $\boldsymbol{P0}$ is NP-complete, and so $\boldsymbol{P0}$ is NP-hard. ■

IV. Latency awarE workloAd offloaDing (LEAD)

We design the LEAD algorithm to efficiently solve the problem, i.e., $\boldsymbol{P0}$. The basic idea of LEAD is to iteratively select a suitable MU i^* ($i^* \in \boldsymbol{\mathcal{I}}$), which generates the maximum workloads among other MUs that are currently not allocated to any cloudlet, i.e.,

$$i^* = \underset{i}{argmax}\left\{\lambda_i \left| \sum_{k \in \boldsymbol{\mathcal{K}}} x_{ik} = 0, i \in \boldsymbol{\mathcal{I}}\right.\right\}. \tag{9}$$

Then, we assign the optimal cloudlet k^* to serve MU i^* (i.e., $x_{i^*k^*} = 1$), which is determined by

$$k^* = \underset{k \in \boldsymbol{\mathcal{K}}}{argmin}\left\{\sum_{j \in \boldsymbol{\mathcal{J}}} y_{ij}t_{jk} + \frac{1}{u_k - \sum\limits_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{ik}} \left| u_k > \sum_{i \in \boldsymbol{\mathcal{I}}}\lambda_i x_{ik}\right.\right\}. \tag{10}$$

That is, the optimal cloudlet corresponds to the one that currently incurs the minimum average response time for serving MU i^* among all the available cloudlets.

LEAD is summarized in Algorithm 1. Specifically, we first initialize the workload offloading assignment matrix $\boldsymbol{\mathcal{X}}$ (i.e., Step1) to be a zero matrix and sort all the MUs in descending order based on their workloads in terms of their average application request generation rates (i.e., Step2). Second, we sequentially select an MU from the sorted MU set and assign the optimal cloudlet to serve the MU based on Eq. 10 (i.e., Step3–Step8). The algorithm terminates when all the MUs are served by corresponding cloudlets. The complexity of LEAD is $O(|\boldsymbol{\mathcal{I}}| \log |\boldsymbol{\mathcal{K}}|)$, where $|\boldsymbol{\mathcal{I}}|$ is the total number of iterations incurred in Algorithm 1 and $\log |\boldsymbol{\mathcal{K}}|$ is the complexity for finding the optimal cloudlet (which incurs the minimum response time) in each iteration.

Algorithm 1 The LEAD algorithm

Input: 1) The location matrix $\boldsymbol{\mathcal{Y}} = \{y_{ij} | i \in \boldsymbol{\mathcal{I}}, j \in \boldsymbol{\mathcal{J}}\}$. 2) The average RTT time matrix $\boldsymbol{\mathcal{T}} = \{t_{jk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\}$. 3) The vector of the average application request generation rate $\boldsymbol{\lambda} = \{\lambda_i | i \in \boldsymbol{\mathcal{I}}\}$. 4) The vector of the average service rate, $\boldsymbol{\mathcal{U}} = \{u_k | k \in \boldsymbol{\mathcal{K}}\}$.

Output: The workload offloading assignment matrix for all the UEs $\boldsymbol{\mathcal{X}} = \{x_{ik} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}}\}$.

```
1: Initialize $\boldsymbol{\mathcal{X}} = \mathbf{0}$.
2: Sort all the MUs in descending order based on their
   average application request generation rates.
3: $i^* = 1$.
4: while $i^* \leq |\boldsymbol{\mathcal{I}}|$ do
5:     Find the optimal cloudlet for MU $i^*$ based on Eq. 10;
6:     Set $x_{i^*k^*} = 1$;
7:     $i^* = i^* + 1$;
8: end while
9: return $\boldsymbol{\mathcal{X}}$.
```

V. Simulation results

We will evaluate the performance of LEAD by comparing it with two other application workload assignment strategies, i.e., the Location awarE workloAd offloadiNg (LEAN) strategy and the remote data center offloading (remoteDC) strategy. The basic idea of LEAN is to offload an MU's requests to the available cloudlet, which is the closest to the MU (i.e., the RTT is the minimum between the MU and the cloudlet). In other words, LEAN only considers the network delay without taking the processing delay into account. On the other hand, remoteDC is to offload all the MUs' requests to the remote data center, which is considered to have sufficient resources. In other words, remoteDC only considers the processing delay without focusing on the network delay.

We apply the MU movement trace provided by the Every-Ware Lab. The trace provides the MUs' movement in the road network of Milan. The whole road network size is 17×28.64 km. There are a total of 100,000 MUs in the area and the location of each MU is monitored for every 20 seconds. Parameters of the MU movement trace are detailed in [10]. We assume the whole area is fully covered by BSs and the coverage size of each BS is 1 km^2; meanwhile, each user will associate with its closest BS for communications and each BS is connected with a dedicated cloudlet for application

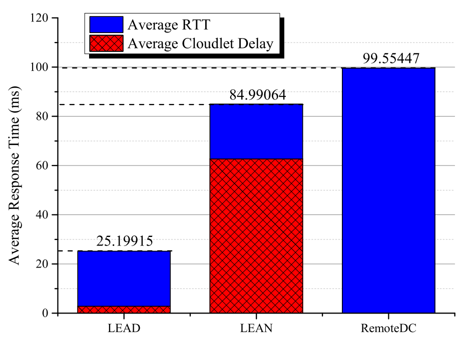

Fig. 2. The average response time among MUs duing the monitoring period ($\bar{\lambda} = 1.76, \bar{u} = 1000, \alpha = 5, \beta = 22.3$).

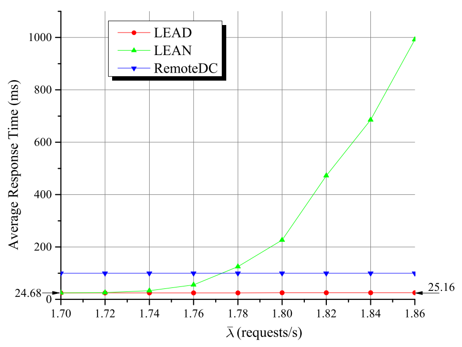

Fig. 3. The average response time delay with respect to the value of $\bar{\lambda}$ ($\bar{u} = 1000, \alpha = 5, \beta = 22.3$).

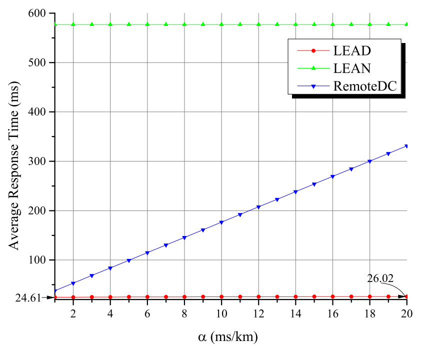

Fig. 4. The average response time with respect to the value of α ($\bar{\lambda} = 1.82, \bar{u} = 1000, \beta = 22.3$).

workload offloading. In addition, we assume that there is a data center deployed in the northeast point of the area. By applying the mentioned MU movement trace, the value of the location matrix for all the MUs, i.e., $\mathcal{Y}$, is obtained.

Each MU's application request generations follow a Poisson distribution, and thus we randomly select the average application request generation rate of each MU between 0 and $\bar{\lambda}$ in each time slot. Meanwhile, we randomly select the average service rate of each cloudlet between 0 and $\bar{u}$. Moreover, we assume the RTT between two different nodes (i.e., a BS and a cloudlet) is a linear function of the distance between the two nodes, i.e., $t_{jk} = \alpha \times d + \beta$, where d is the distance between BS j and cloudlet k, and α and β are the coefficients.

The length of each time slot is 5 minutes and we monitor the response time of the offloading process for each MU for two hours. Fig. 2 shows the average response time for all the MUs during the period. Obviously, LEAD produces the lowest average response time than the others. Specifically, LEAN and LEAD generates the similar average RTT, but LEAN always assign MUs' application workloads to their closest available cloudlets, which may be heavily loaded. Thus, LEAN incurs higher average processing delay than LEAD. On the other hand, remoteDC has sufficient computing resources, and so the average cloudlet delay is negligible; however, remoteDC incurs higher average RTT, which results in its average response time much worse than LEAD.

We further analyze how the application workloads of MUs affect the performance of the three strategies. Note that the value of $\bar{\lambda}$ reflects the application workloads of MUs, i.e., increasing the value of $\bar{\lambda}$ increases the application workloads from all the MUs. As shown in Fig. 3, when $\bar{\lambda}$ becomes larger, the average response time of LEAN increases exponentially while the performances of LEAD and remoteDC are not affected by $\bar{\lambda}$. This is because LEAD can offload the MU's workloads to the lightly loaded cloudlet to minimize the total delay. The data center is considered to have sufficient resources to handle the workloads from MUs, and so increasing the application workloads cannot significantly increase the response time of remoteDC, which is mainly determined by the RTT between MUs and the data center.

In addition, we analyze how the network traffic load affects the performance of the three strategies. Note that if the network traffic load is increasing, the average RTT between two different nodes becomes longer. We use the value of α to indicate the network traffic load, i.e., larger value of α means one unit of distance is mapped into longer RRT. As shown in Fig. 4, when the value of α becomes larger, the average delay of remoteDC increases linearly. Yet, those of LEAD and LEAN do not significantly change. This is because as the network traffic load is increasing, LEAD and LEAN select the closest available cloudlet to process the MU's workloads.

VI. Conclusion

In this paper, we have proposed the cloudlet network to bring the computing resource to the mobile edge. We have also designed the LEAD strategy in the context of the cloudlet network to allocate MUs' application workloads into suitable cloudlets such that the average response time of the application workload offloading process among MUs is minimized.

References

[1] X. Sun and N. Ansari, "EdgeIoT: Mobile Edge Computing for the Internet of Things," *IEEE Comm. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.
[2] X. Sun, N. Ansari, and Q. Fan, "Green Energy Aware Avatar Migration Strategy in Green Cloudlet Networks," in *IEEE 7th Intl. Conf. on Cloud Comp. Technology & Science*, Vancouver, BC, 2015, pp. 139–146.
[3] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Trans. on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, 2016.
[4] A. Mukherjee, D. De, and D.G. Roy, "A Power and Latency Aware Cloudlet Selection Strategy for Multi-Cloudlet Environment," in *IEEE Trans. on Cloud Comp.*, doi: 10.1109/TCC.2016.2586061, *early access*.
[5] X. Sun and N. Ansari,"PRIMAL: PRofIt Maximization Avatar pLacement for Mobile Edge Computing," in *Proc. 2016 IEEE Intl. Conf. Comm.*, Kuala Lumpur, Malaysia, May 23–27, 2016, pp. 1–6.
[6] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *35th Annual IEEE Intl. Conf. on Comp. Comm.*, San Francisco, CA, 2016, pp. 1–9.
[7] K. Elgazzar, P. Martin, and H. S. Hassanein, "Cloud-Assisted Computation Offloading to Support Mobile Services," *IEEE Trans. on Cloud Computing*, vol. 4, no. 3, pp. 279–292, July–Sept. 1 2016.
[8] C. Yu, *et al.*, "Software-Defined Latency Monitoring in Data Center Networks," in *Intl Conf. on Passive and Active Measurement*, New York City, NY, Mar. 19–20, 2015. pp. 360–372.
[9] T.G. Rodrigues, *et al.*, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing through VM Migration and Transmission Power Control," *IEEE Trans. on Computers*, doi: 10.1109/TC.2016.2620469, *early access*.
[10] User Movement Simulations Project. Available. [Online]: http://everywarelab.di.unimi.it/lbs-datasim.