Application Aware Workload Allocation for Edge Computing based IoT

Citation:
Q. Fan and N. Ansari, "Application Aware Workload Allocation for Edge Computing-Based IoT," in *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146-2153, June 2018.

URL:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8336866

Application Aware Workload Allocation for Edge Computing based IoT

Qiang Fan, *Student Member, IEEE* and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—Empowered by computing resources at the network edge, data sensed from IoT devices can be processed and stored in their nearby cloudlets to reduce the traffic load in the core network, while various IoT applications can be run in cloudlets to reduce the response time between IoT users (e.g., user equipment in mobile networks) and cloudlets. Considering the spatial and temporal dynamics of each application's workloads among cloudlets, the workload allocation among cloudlets for each IoT application affects the response time of the application's requests. While assigning IoT users' requests to their nearby cloudlets can minimize the network delay, the computing delay of a type of requests may be unbearable if the corresponding virtual machine of the application in a cloudlet is overloaded. To solve this problem, we design an Application awaRE workload Allocation (AREA) scheme for edge computing based IoT to minimize the response time of IoT application requests by deciding the destination cloudlets for each IoT user's different types of requests and the amount of computing resources allocated for each application in each cloudlet. In this scheme, both the network delay and computing delay are taken into account, i.e., IoT users' requests are more likely assigned to closer and lightly loaded cloudlets. Meanwhile, the scheme will dynamically adjust computing resources of different applications in each cloudlet based on their workloads, thus reducing the computing delay of all requests in the cloudlet. The performance of the proposed scheme has been validated by extensive simulations.**

***Index Terms*—Cloudlet, Internet of Things (IoT), workload allocation, edge computing, resource allocation.**

I. Introduction

In the past few years, a tremendous number of smart devices and objects, such as smart phones, wearable devices, industrial and utility components, are equipped with sensors to sense the real-time physical information from the environment [1]. Hence, Internet of Things (IoT) is introduced, where various smart devices are connected with each other via the internet and empowered with data analytics. Various IoT applications, such as smart transportation, smart health, smart city and smart home have been widely studied to improve our daily life. Owing to the high volume and fast velocity of data streams generated by IoT devices, the cloud that can provision flexible and efficient computing resources is employed as a smart "brain" to process and store the big data generated from distributed IoT devices [2]. However, as the data streams generated from IoT devices are transmitted to the remote cloud via Internet, the transferred data may consume a huge amount of bandwidth and energy of the core network. On the other hand, since the remote cloud is far from IoT users which send application requests and await the results generated by the data processing in the remote cloud, the response time of the requests may be too long, especially unbearable for delay sensitive IoT applications. Therefore, cloudlets, which bring computing resources close to IoT devices and IoT users, can be employed to alleviate the traffic load in the core network and minimize the response time for IoT users [3], [4].

Although provisioning cloudlets may reduce the network delay, simply allocating all IoT users' workloads to the closest cloudlets is not enough to reduce the response time, which consists of both the network delay and computing delay. Note that we consider UE equipments (UEs) in mobile networks as IoT users, each of which can run several types of IoT Apps. Owing to the UE mobility in the network, the workload distribution exhibits spatial and temporal dynamics among cloudlets. When the workload of a cloudlet is too heavy, the computing resources available for an application is limited, and thus the response time of the corresponding requests is degraded correspondingly. In this case, although the cloudlet in the proximity yields the minimum network delay, the bulk of the response time is attributed to the computing delay. Thus, the workload allocation of different types of requests greatly impacts the response time of UEs' requests. On the other hand, for each cloudlet, the resource allocation for different types of applications also affects the computing delay of different types of requests. Since the computing size per request for different applications are different, the computing capacity of a cloudlet should be optimally allocated for different types of applications in order to reduce the computing delay of all Apps of UEs.

To solve the above problem, we propose an Application awaRE workload Allocation (AREA) scheme for edge computing based IoT to minimize the total response time of UEs' Apps, where both the network delay and computing delay are taken into account. Below are major contributions of the paper.

- We formulate the problem of minimizing the average response time of different types of IoT Apps by offloading UEs' different types of requests among distributed cloudlets and allocating optimal computing resources for different applications in each cloudlet. The response time of each type of requests consists of both the network delay and computing delay. To reduce the network delay, different types of requests of a UE are favorably assigned to closer cloudlets. On the other hand, each application is assumed to be handled by a dedicated virtual machine in each cloudlet,

Q. Fan and N. Ansari are with Advanced Networking Lab., Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102 USA (Email: {qf4, nirwan.ansari}@njit.edu). This work was supported in part by the National Science Foundation (CNS-1647170).

the capacity of which can be dynamically allocated in each time slot [5]; when a cloudlet is overloaded, the computing resources available for each application are not enough to handle the type of requests, and thus the computing delay becomes the dominating factor of the response time. Hence, different types of requests of a UE should be assigned to other lightly loaded cloudlets to reduce their computing delay.

- Since different applications require different QoS constraints in terms of the maximum allowed computing delay, we will allocate different types of requests of each UE to suitable cloudlets to guarantee that their corresponding QoS constraints are satisfied.
- To solve the application aware workload allocation problem in each time slot, we design the novel AREA algorithm that decomposes the problem into two sub-problems and solve them sequentially. First, we design a heuristic algorithm to allocate each UE's different types of requests to suitable cloudlets according to the workloads of cloudlets and the network delays between the UE and cloudlets. Afterwards, when different types of requests of each UE have been assigned among cloudlets, the above problem becomes a resource allocation problem in each cloudlet that has been proved to be a convex problem, and thus we can optimally allocate the computing resources of each cloudlet to different types of applications by solving the convex problem.

The remainder of this paper is organized as follows. In Section II, we briefly review related works. In Section III, we illustrate the cloudlet network architecture and describe the system model. In Section IV, we formulate and analyze the application aware workload allocation problem. In Section V, the AREA algorithm is proposed to obtain the suboptimal solution of the application aware workload allocation problem. Section VI shows the simulation results, and concluding remarks are presented in Section VII.

II. Related Works

Mobile edge computing, by moving computing resources close to UEs, has been proposed to improve UE experience for mobile applications. Tong *et al.* [6] proposed a workload placement algorithm in a hierarchical edge cloud network, which selects the cloudlet and allocates the computing resources for each task to minimize the response time for all offloaded tasks. Fan *et al.* [7] proposed to migrate UEs' virtual machines (VMs) among distributed cloudlets to reduce the brown energy consumption of cloudlets by considering the green energy generation among cloudlets and energy consumption of VM migrations. Some works [8], [9] look into placing a certain number of cloudlets among a given set of available sites and then assigning workloads to the cloudlets.

Owing to the proximity of edge computing resources to IoT devices and IoT users, some studies have focused on integrating IoT with mobile edge computing. Chiang *et al.* [10] summarized the opportunities and challenges of edge computing in the networking context of IoT and advocated that edge computing can fill the technology gaps in IoT. Sun and Ansari [11] proposed the IoT architecture (EdgeIoT) to handle the data streams from IoT devices at the mobile edge. Moreover, Jutila [12] proposed adaptive edge computing solutions for IoT networking in order to optimize traffic flows and network resources. Deng *et al.* [13] proposed an algorithm to balance the power consumption and service delay by allocating workloads among fog nodes and the cloud. Yousefpour *et al.* [14] proposed a delay aware policy for the IoT-fog-cloud network to minimize the service delay for IoT applications. Zhang *et al.* [15] proposed an edge IoT framework to allocate the limited computing resources of fog nodes to IoT users to achieve the optimal and stable performance in the IoT based network. Fan *et al.* [16] proposed a workload allocation scheme, referred to as WALL, in a hierarchical edge network to optimize the response time of task requests. Jia *et al.* [17] proposed to place a certain number of cloudlets and allocate workloads among cloudlets to minimize the response time. Yang *et al.* [18] studied the joint optimization of application service placement and load dispatching among cloudlets where all users' workloads are the same, and then designed a set of efficient algorithms to achieve various trade-offs among the average latency of users' requests and the cost of service providers. As we know, the computing size per request for different applications is heterogeneous while their QoS requirements are different. However, all the above works assume that application requests are homogeneous and then allocate the workloads among cloudlets to minimize the response time of requests. Considering the diverse computing sizes and QoS requirements of different types of requests, we formulate the problem of minimizing the average response time of different types of Apps by assigning UEs' different types of Apps to distributed cloudlets and allocating optimal computing resources for different applications in each cloudlet. The problem is formulated such that the QoS constraint of each application in terms of the maximum allowable computing delay is satisfied individually.

III. System Model

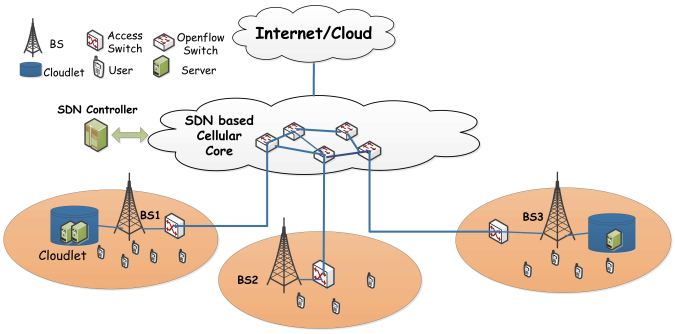

Fig. 1. Cloudlet network architecture.

A distributed cloudlet network architecture is illustrated in Fig. 1, where cloudlets are co-located with some base stations (BSs). The software defined network (SDN), which consists of a SDN controller and open flow switches, is employed as the cellular core network, thus enabling flexible routing and communications resource among BSs. All BSs are equipped with two interfaces (i.e., NB-IoT and LTE) to offer

the seamless coverage for both IoT devices and IoT users (UEs). Thus, the sensed data of IoT devices can be stored at their closest cloudlets and the remote cloud, which act as brokers. Meanwhile, a Resource Directory (RD) is located at the SDN controller to help each IoT application discover the location of its required IoT data. On the other hand, each UE can access different cloudlets through its BS and the SDN based cellular core network. Within one cloudlet, we assume that each virtual machine (VM) only processes the workloads of one application, i.e., each application is mapped to a dedicated VM. Note that each IoT application has only one VM in a cloudlet. Considering the diversity of applications, the computing capacities of VMs are heterogeneous in a cloudlet and can be adjusted dynamically [5]. We define an IoT App as the software program running on a UE that requests the specific type of application service. As a UE may run multiple IoT Apps, each type of application requests of the UE can be offloaded to a cloudlet having the corresponding type of VMs. Thus, when an application VM in a cloudlet receives an application request, it quickly retrieves the required IoT data from other brokers under the direction of RD and then processes the request to get the result.

TABLE I
LIST OF SYMBOLS

Symbol	Definition
$\mathcal{I}$	Set of distributed cloudlets.
$\mathcal{J}$	Set of UEs.
$\mathcal{K}$	Set of different IoT applications.
$\mathcal{R}$	Set of BSs.
x_{ijk}	Binary indicator of UE j's App k being assigned to cloudlet i.
y_{rj}	Binary indicator of UE j being covered by BS r.
$\mathcal{K}_j$	Set of Apps run by UE j.
μ_{ik}	Computing capacity of type k VM in cloudlet i.
τ_{ri}	E2E delay between BS r and cloudlet i.
λ_{jk}	Average request arrival rate of type-k App in UE j.
λ_{ik}	Average request arrival rate of type k VM in cloudlet i.
l_k	Average computing size of a type-k request.
d_{ij}	Network delay between UE j and cloudlet i.
D_k	Maximum allowed computing delay of Application k.
$\mathcal{Z}$	Set of Apps of all UEs.
j_z	Index of the UE where App $z \in Z$ is located.
d_{iz}	Network delay between App z and cloudlet i.

Note that each UE may have several types of IoT Apps. As each App in a UE is assigned to only one cloudlet individually, the size of the set of Apps in the network can be derived as: $|\mathcal{Z}| = \sum_{j \in \mathcal{J}} |\mathcal{K}_j|$, in which the variables are defined in the list of symbols shown in Table I.

A. Computing Delay

Assume that type k requests of UE j are generated according to a Poisson Process with the average arrival rate λ_{jk}. Thus, the workload of type k VM in cloudlet i can be expressed as:

$$\lambda_{ik} = \sum_{j \in J} x_{ijk} \lambda_{jk}, \tag{1}$$

and it also follows a Poisson Process. On the other hand, the computing capacity (in terms of CPU cycles per second) of type k VM in cloudlet i (i.e., μ_{ik}) is fixed in each time slot; the computing size of a type k application request (in terms of the CPU cycles) follows an exponential distribution with the average value of l_k. Thus, we can derive the service time for type k requests running in a cloudlet's VM as l_k/μ_{ik}, which also follows an exponential distribution. Since the arrival rate of each VM of a cloudlet follows a Poisson Process while the corresponding service time follows an exponential distribution, each VM of a cloudlet can form an M/M/1 queuing model to process its corresponding application requests. Note that to keep the queue stable, the average arrival rate of the VM (i.e., λ_{ik}) should be smaller than its average service rate (i.e., μ_{ik}/l_k), and thus we can derive that $\mu_{ik}/l_k - \lambda_{ik} > 0$. We define the computing delay of type k requests in cloudlet i, t_{ik}, as the average system delay of type k VM's queue (i.e., including the waiting delay and service time):

$$t_{ik} = \frac{1}{\mu_{ik}/l_k - \sum_{j \in J} x_{ijk} \lambda_{jk}}, \forall i \in \mathcal{I}, k \in \mathcal{K}. \tag{2}$$

B. Network Delay

When a request of a UE is sent to a cloudlet, the request goes through its BS and the SDN-based cellular core network. Therefore, the E2E delay between a UE's App and its cloudlet consists of two parts: first, the E2E delay between the UE and its associated BS, i.e., the wireless delay; second, the E2E delay between its BS and its assigned cloudlet. However, the cloudlet selection for a UE does not affect its wireless delay, which only depends on the UE's service plan and the mobile provider's bandwidth allocation strategy [19]. Thus, we just consider the E2E delay between the BS and cloudlet in this paper. Denote τ_{ri} as the E2E delay between BS r and cloudlet i, and $\mathcal{Y}$ as a given indicator matrix to reflect the UE-BS association at the beginning of each time slot, in which $y_{rj} \in \mathcal{Y}$ represents whether UE j is covered by BS r or not. Note that the value of τ_{ri} can be measured and recorded by the SDN controller [20], [21]. Thus, the network delay between UE j and cloudlet $i \in \mathcal{I}$ can be expressed as

$$d_{ij} = \sum_{r \in \mathcal{R}} y_{rj} \tau_{ri}, \ \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{3}$$

IV. PROBLEM FORMULATION

The response time of a UE's App consists of both the computing delay and network delay. In the workload allocation, both of them should be taken into account. On one hand, owing to the dynamic distribution of workloads among different cloudlets, the overloaded cloudlets incur remarkably higher computing delay than other lightly loaded cloudlets. Thus, if the closest cloudlet of a UE is overloaded, the requests of each App of the UE should be allocated to alternative cloudlets to reduce the response time. On the other hand, offloading an App's requests from its closest cloudlet to other cloudlets will increase the network delay. The main goal of this paper is to minimize the response time of all IoT Apps in the network by assigning the requests of each App among cloudlets and

flexibly allocating the computing resource of each cloudlet to different types of VMs to serve the assigned Apps. Thus, we can formulate the application aware workload allocation problem in each time slot as follows:

$$P1: \min_{x_{ijk},\mu_{ik}} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_j} x_{ijk}\left(d_{ij}+\frac{1}{\mu_{ik}/l_k-\sum\limits_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}}\right) \tag{4}$$

$$s.t. \sum_{k\in\mathcal{K}} \mu_{ik} \le C_i, \forall i\in\mathcal{I}, \tag{5}$$

$$\sum_{i\in\mathcal{I}} x_{ijk}=1, \forall j\in\mathcal{J}, \forall k\in\mathcal{K}_j, \tag{6}$$

$$x_{ijk}\left(\frac{1}{\mu_{ik}/l_k-\sum\limits_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}}\right) \le x_{ijk}D_k, \tag{7}$$

$$\forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall k\in\mathcal{K}_j,$$

$$\mu_{ik}/l_k-\sum_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}>0, \forall i\in I, \forall k\in\mathcal{K}, \tag{8}$$

$$x_{ijk}\in\{0,1\}, \forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall k\in\mathcal{K}_j, \tag{9}$$

$$\mu_{ik}\in[0,C_i], \forall i\in\mathcal{I}, \forall k\in\mathcal{K}. \tag{10}$$

Here, C_i is the computing capacity of cloudlet i and D_k is the maximum allowed computing delay of application k. Constraint (5) indicates that the aggregated computing resources of all VMs in a cloudlet should be no larger than the cloudlet's computing capacity. Constraint (6) ensures that each App of a UE is assigned to only one cloudlet. Constraint (7) imposes the computing delay for each UE's type k APP to meet the QoS requirement of the application in terms of the maximum allowed computing delay D_k. Constraint (8) imposes the average service rate of VM k in a cloudlet to be smaller than the VM's average task arrival rate, in order to keep the queue of the VM stable.

Lemma 1. *The problem of application aware workload allocation (i.e., P1) is NP-hard.*

Proof: Suppose there is only one IoT application; the capacity of VM k equals to the capacity of a cloudlet, i.e., $\mu_{ik}=C_i$. Meanwhile, we assume that the computing delay threshold $D_k=+\infty$. Therefore, both Constraint (5) and (7) can be relaxed from P1. Then, to prove that P1 is a NP-hard problem, we can demonstrate that its corresponding decision problem is NP-complete. The decision problem of P1 can be expressed as: given a positive value of b, is it possible to find a feasible solution $X=\{x_{ijk}|i\in\mathcal{I}, j\in\mathcal{J}\}$ such that $\sum\limits_{i\in I}\sum\limits_{j\in\mathcal{J}} x_{ijk}\left(d_{ij}+\frac{1}{\mu_{ik}/l_k-\sum\limits_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}}\right)\le b$, and Constraints (6), (8) and (9) are satisfied?

In order to prove that the above decision problem is NP-complete, only two cloudlets are considered and the average service rate of either cloudlet is set to be the same, i.e., $\mu_1/l_k=\mu_2/l_k=\frac{1}{2}\sum\limits_{j\in\mathcal{J}}\lambda_{jk}+\epsilon$, where ϵ is a very small positive value, i.e., $\epsilon\ll\frac{1}{2}\min\{\lambda_{jk}|j\in\mathcal{J}\}$. Moreover, assume that $b\to+\infty$. Thus, $\sum\limits_{i\in\mathcal{I}}\sum\limits_{j\in\mathcal{J}} x_{ijk}\left(d_{ij}+\frac{1}{\mu_{ik}/l_k-\sum\limits_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}}\right)\le b$ is always satisfied for all solutions of $\mathcal{X}$ and can be relaxed. To satisfy Constraint (8) (i.e., $\mu_{ik}/l_k-\sum\limits_{j\in\mathcal{J}} x_{ijk}\lambda_{jk}>0, \forall i\in\mathcal{I}$), we need to guarantee that $\sum\limits_{j\in\mathcal{J}}\lambda_{jk}x_{1jk}=\sum\limits_{j\in\mathcal{J}}\lambda_{jk}x_{2jk}=\frac{1}{2}\sum\limits_{j\in\mathcal{J}}\lambda_{jk}$. Consequently, the decision problem can be transformed into a partition problem, i.e., whether the UEs can be partitioned into two sets to make the average request arrival rates of the two sets the same. Hence, the partition problem is reducible to the decision problem of P1. As the partition problem is a well-known NP-complete problem, the decision problem of P1 is also NP-complete, and thus P1 is NP-hard. ■

V. The AREA Algorithm

Since P1 is NP-hard, which is challenging to achieve the optimal solution, we propose the heuristic AREA algorithm to effectively allocate different types of workloads among cloudlets as well as flexibly allocate computing resources for different VMs in each cloudlet, with low computational complexity. Note that the major challenge of solving P1 is that μ_{ik} depends on the App assignment x_{ijk}. To solve P1 more efficiently, we decompose the original problem into two sub-problems: the App assignment sub-problem and the resource allocation sub-problem. We will first assign different types of Apps among cloudlets (i.e., determining x_{ijk}), and then try to optimally allocate the computing resources to different types of VMs in each cloudlet (i.e., μ_{ik}) based on the given x_{ijk}.

A. App Assignment

When assigning Apps' workloads among cloudlets, the priority of assigning each App to its closest cloudlets should be considered to reduce the total network delay. Therefore, we will initialize the App assignment by allocating all Apps to their closest cloudlets; then, the algorithm will iteratively select a suitable App with the highest response time and reallocate it to an alternative cloudlet which minimizes its response time, until each App cannot find a better cloudlet.

Given the capacities of cloudlets, the initial App assignment is determined by the network delay between UEs that host Apps and cloudlets, and thus can be obtained by solving the following problem:

$$P2: \min_{x_{ijk}} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_j} x_{ijk}d_{ij} \tag{11}$$

$$s.t. \sum_{i\in\mathcal{I}} x_{ijk}=1, \forall j\in\mathcal{J}, \forall k\in\mathcal{K}_j, \tag{12}$$

$$\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_j}\lambda_{jk}l_k x_{ijk}\le C_i, \forall i\in\mathcal{I} \tag{13}$$

$$x_{ijk}\in\{0,1\}, \forall i\in\mathcal{I}, \forall j\in\mathcal{J}, \forall k\in\mathcal{K}. \tag{14}$$

As each App of a UE is assigned among cloudlets individually, we denote $\mathcal{Z}_1$ as the set of Apps of all UEs which are waiting to be assigned among cloudlets, and $\mathcal{I}_1$ as the set of cloudlets which have excess computing resources. At the beginning, all UEs' Apps have not be assigned and are

included in $\mathcal{Z}_1$ (i.e., $\mathcal{Z}_1 = \mathcal{Z}$), while all cloudlets are empty without any assigned Apps, i.e., all cloudlets are included in $\mathcal{I}_1$. Denote d_{iz} as the network delay between an App z (i.e., $z \in \mathcal{Z}_1$) and cloudlet i, j_z as the UE where App z is located. Hence, we have $d_{iz} = d_{ij_z}, \forall i \in \mathcal{I}, \forall z \in \mathcal{Z}_1$.

In the initialization, for App z, the optimal cloudlet $i^* \in \mathcal{I}_1$ is the one that incurs the lowest network delay, i.e., $i^* = \arg\min\{d_{iz} | i \in \mathcal{I}_1\}$; the suboptimal cloudlet $i^{'}$ is the one that incurs the second lowest network delay among the cloudlets in $\mathcal{I}_1$, i.e., $i^{'} = \arg\min_{i}\{d_{iz} | i \in \{\mathcal{I}_1 \backslash i^*\}\}$.

As shown in P2, the capacity of each cloudlet is limited, and thus it is impossible to allocate all Apps to their corresponding optimal cloudlets. The basic idea of the initialization is to iteratively select a suitable App, whose suboptimal cloudlet $i^{'}$ incurs a significant network delay degradation as compared to the optimal cloudlet i^*, and then allocate the App into its optimal cloudlet. It is easy to observe that the network delay degradation incurred by the suboptimal cloudlet determines the priority of assigning App z to its optimal cloudlet. For example, if App z's suboptimal cloudlet B leads to a remarkably higher delay than its optimal cloudlet A as compared to other Apps, assigning App z to the suboptimal cloudlet will significantly impact the total network delay of all Apps. In this case, App z is given a higher priority than other Apps to be assigned into its optimal cloudlet A.

Denote Δd_z as the network delay degradation by allocating App z from the optimal cloudlet i^* to the suboptimal cloudlet $i^{'}$, i.e.,

$$\Delta d_z = d_{i^{'}z} - d_{i^*z}, \forall z \in \mathcal{Z}_1. \tag{15}$$

Thus, as shown in Algorithm 1, in each iteration of the initialization, the algorithm will select and allocate a suitable App z, which has the highest network delay degradation (i.e., $z = \arg\max_{z}\{\Delta d_z | z \in \mathcal{Z}_1\}$), to its optimal cloudlet. Afterwards, if the workload of a cloudlet exceeds its capacity, the cloudlet is removed from $\mathcal{I}_1$. Note that once $\mathcal{I}_1$ is updated, the algorithm has to recalculate i^*, $i^{'}$ and Δd_z for each App $z \in \mathcal{Z}_1$. The above procedure is repeated until all Apps are assigned among cloudlets, i.e., $\mathcal{Z}_1 = \emptyset$.

Lemma 2. *Algorithm 1 terminates after a finite number of iterations, yielding a feasible IoT App assignment.*

Proof: Let $\xi = |\mathcal{I}_1| = N$ initially, i.e., $\xi > 0$. Then, for each iteration, since the algorithm chooses a suitable App z, where $z = \arg\max_{z}\{\Delta d_z | z \in \mathcal{Z}_1\}$, and allocates it to its optimal cloudlet i^* (i.e., $i^* = \arg\min_{i}\{d_{iz} | i \in \mathcal{I}_1\}$), ξ is decremented by one. As a result, ξ will become zero after a finite number of iterations, and thus $\mathcal{I}_1 = \emptyset$. ■

As shown in Algorithm 1, the complexity of Step 2 is $|\mathcal{Z}|$. After Step 2, the complexity of Steps 4-5 is $O(|\mathcal{Z}| + |\mathcal{I}|)$ in the worst case; as they repeat for $|\mathcal{Z}|$ times, the corresponding complexity is $O(|\mathcal{Z}|(|\mathcal{Z}| + |\mathcal{I}|))$. Meanwhile, as Steps 9-10 repeat for at most $|\mathcal{I}|$ times, the corresponding complexity is $O((|\mathcal{Z}| + 1)|\mathcal{I}|)$. Aggregating all these steps, the complexity of Algorithm 1 becomes $O(|\mathcal{Z}|(|\mathcal{Z}| + |\mathcal{I}|))$.

After the initialization, the AREA algorithm, as shown in Algorithm 2, iteratively selects a suitable App with the highest response time, and reallocates it to an alternative cloudlet. At the beginning, all Apps are unmarked and we define $\mathcal{Z}_2$ as the set of unmarked Apps. Then, in each iteration, the AREA algorithm finds the App with the highest response time among all unsigned Apps, and searches for a new cloudlet for the App to minimize its response time. Note that in each iteration, the computing resource for each application in a cloudlet is determined by the percentage of the application's workload in the total workloads in the cloudlet, and thus we can derive the response time of Apps in different cloudlets. If a new cloudlet is found, AREA proceeds to the next iteration. Otherwise, the algorithm marks the App (i.e., removing the App from $\mathcal{Z}_2$) and continues to the next iteration. The AREA algorithm repeats the iterations until $\mathcal{Z}_2 = \emptyset$.

Algorithm 1

Input: The UE-BS association vector $\mathcal{Y} = \{y_{rj} | r \in \mathcal{R},\ j \in \mathcal{I}\}$. The matrix of E2E delay between BSs and cloudlets $\mathcal{T} = \{\tau_{ri} | r \in \mathcal{R},\ i \in \mathcal{I}\}$. The vector of the average task arrival rate for UEs' Apps $\Lambda = \{\lambda_{jk} | j \in \mathcal{J}, j \in \mathcal{K}_j\}$.

Output: The initial App assignment matrix, i.e., $\mathcal{X} = \{x_{ijk} | i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}_j\}$.

1: Set $\mathcal{Z}_1 = \mathcal{Z}$ and $\mathcal{I}_1 = \mathcal{I}$ based on their definitions;
2: $\forall z \in \mathcal{Z}_1$, calculate Δd_z based on Eq. (15);
3: **while** $\mathcal{Z}_1 \neq \emptyset$ **do**
4: Find App z, where $z = \arg\max_{z}\{\Delta d_z | z \in \mathcal{Z}_1\}$;
5: Allocate App z to its optimal cloudlet i^* (i.e., $i^* = \arg\min_{i}\{d_{ij} | i \in \mathcal{I}_1\}$);
6: Let $x_{ij_zk_z} = 1$;
7: Update the App set $\mathcal{Z}_1$, i.e., $\mathcal{Z}_1 = \mathcal{Z}_1 \backslash z$.
8: **if** cloudlet i^* is full **then**
9: Remove i^* from $\mathcal{I}_1$, i.e., $\mathcal{I}_1 = \mathcal{I}_1 \backslash i^*$;
10: $\forall z \in \mathcal{Z}_1$, recalcuate Δd_z based on Eq. (15);
11: **end if**
12: **end while**
13: **return** $\mathcal{X}$.

We now analyze the computational complexity of Algorithm 2. In each iteration, the algorithm checks cloudlets for an App, and the number of related cloudlets can be $|\mathcal{I}|$ in the worst case. Therefore, the complexity of each iteration is $O(|\mathcal{I}|)$. Then, we analyze the required number of iterations for the algorithm to optimally place all Apps among the cloudlets. Each App has a choice of up to $|\mathcal{I}|$ cloudlets. In each cloudlet, the App can have at most $|\mathcal{Z}|$ different response times owing to the different number of Apps allocated to the cloudlet. As a result, the number of improvements for the App is limited by $|\mathcal{I}||\mathcal{Z}|$. Thus, considering the number of Apps is $|\mathcal{Z}|$, the total number of iterations in the worst case is $|\mathcal{I}||\mathcal{Z}|^2$. So, the computational complexity of Algorithm 2 is $O(|\mathcal{I}|^2|\mathcal{Z}|^2)$. When we fix the number of cloudlets $|\mathcal{I}|$, the complexity of Algorithm 2 is polynomial with respect to the number of the Apps.

B. Resource Allocation

After all UEs' Apps are assigned to different cloudlets, the problem can be transformed into a resource allocation problem

Algorithm 2

Initialize App assignment by Algorithm 1 and obtain $\mathcal{X}$;
Set $\mathcal{Z}_2$ based on its definition, i.e., $\mathcal{Z}_2 = \{z | z \in \mathcal{Z}\}$
while $\mathcal{Z}_2 \neq \emptyset$ **do**
  Find App $z \in \mathcal{Z}_2$ with the highest response time;
  Obtain the current cloudlet i of App z;
  Find the suitable cloudlet i^* for App z, i.e., $i^* = \arg\min \left(d_{ij} + \frac{1}{\mu_{ik}/l_k - \sum_{j \in \mathcal{J}} x_{ijk}\lambda_{jk}} \right)$;
  if $i^* \neq i$ **then**
    Assign App z to the new cloudlet i^* and update $\mathcal{X}$;
  else
    Mark App z and let $\mathcal{Z}_2 = \mathcal{Z}_2 \backslash z$;
  end if
end while
return $\mathcal{X}$.

for each cloudlet i as follows:

$$P3: \min_{\mu_{ik}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} x_{ijk} \left(d_{ij} + \frac{1}{\mu_{ik}/l_k - \sum_{j \in \mathcal{J}} x_{ijk}\lambda_{jk}} \right) \quad (16)$$
$$s.t. \quad Constraints(5), (7), (8), (10).$$

We can then prove the following lemma:

Lemma 3. *When each x_{ijk} is determined, $P3$ is a convex optimization problem.*

Proof: For brevity, let $f = \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} x_{ijk} \left(d_{ij} + \frac{1}{\mu_{ik}/l_k - \sum_{j \in \mathcal{J}} x_{ijk}\lambda_{jk}} \right)$, and we use μ_k to substitue μ_{ik} in cloudlet i. Thus, we have

$$\frac{\partial^2 f}{\partial \mu_k \partial \mu_{k'}} = \begin{cases} \sum_{j \in \mathcal{J}} 2 x_{ijk} l_k^{-2} (\mu_k/l_k - \sum_{j \in \mathcal{J}} x_{ijk}\lambda_{jk})^{-3}, \text{if} k = k', \\ 0, \text{ otherwise.} \end{cases} \quad (17)$$

Since $(\mu_k/l_k - \sum_{j \in \mathcal{J}} x_{ijk}\lambda_{jk}) > 0$, the Hessian matrix $H = \frac{\partial^2 f}{\partial \mu_k \partial \mu_{k'}}$ of f is a positive definite matrix. As a result, function f is convex. Moreover, because Constraints (5), (7), (8), (10) are linear, the optimization problem $P3$ is a convex optimization problem. ■

As $P3$ is a convex problem, we can derive the optimal solution of $P3$ by solving the KKT condition of $P3$ [22]. Therefore, the computing resource of each cloudlet is optimally allocated to different VMs to minimize the response time. Consequently, the suboptimal solution of P1 is achieved.

VI. Results and Discussion

In this section, we set up simulations of the proposed scheme to evaluate its performance. We select two other workload allocation strategies for comparison: the density-based clustering (DBC) strategy [17] and the latency-based strategy [18]. The basic idea of DBC is to offload UEs' workloads to suitable cloudlets until the workloads of the cloudlets exceed the average workload among cloudlets. On the other hand, the latency-based strategy is to minimize the network delay between Apps and cloudlets by assigning Apps to suitable cloudlets. In the above two strategies, the computing resource of each cloudlet is allocated to different types of VMs according to the percentage of different types of workloads in the cloudlet.

The simulation environment consists of 25 BSs within an area of 25 km^2, where the coverage of each BS is 1 km^2 and each BS is attached with a cloudlet. Meanwhile, 1000 UEs are uniformly distributed among the BSs and assumed to be associated with their closest BSs. There are 10 types of IoT applications in the cloudlet network, and we randomly choose 3 types of Apps for each UE (i.e., the total number of Apps in the network is 3000). The length of each time slot is set as 5 mins. As each App's task arrival rate follows a Poisson distribution, we randomly choose the average task arrival rate of each App between 0 and λ_{max}. As the computing sizes of application k's requests follow an exponential distribution with the average value of l_k, the average size of different types of requests is chosen according to the Normal distribution with an average of 10^6 CPU cycles and a variance of $2*10^5$ cycles, i.e., $N(10^6, 2*10^5)$. Moreover, we assume the network delay between a BS and a cloudlet is a linear function of the distance between them [4], [23], i.e., $\tau_{ri} = \alpha \times d + \beta$, where d is the distance between BS r and cloudlet i, and α and β are set as 5 and 22.3, respectively. In addition, the maximum allowed computing delay for different types of applications is chosen according to $N(60, 20)$ (ms).

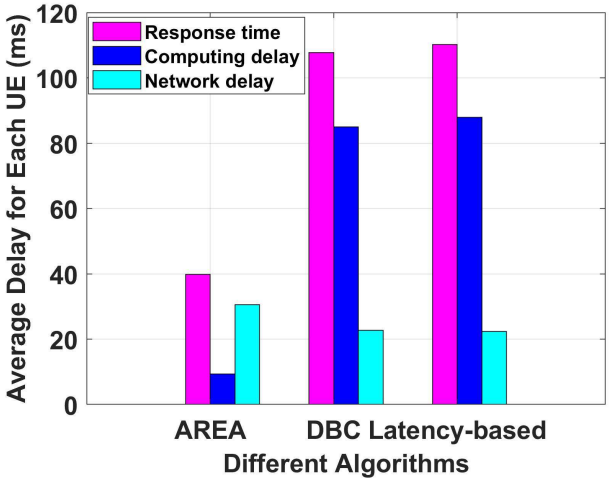

Fig. 2. Average performance of an App for different algorithms ($\lambda_{max} = 1.5$, $C_i = 2 * 10^8$).

Fig. 2 shows the average response time per App, in which AREA achieves lower response time as compared to the other two strategies. Specifically, the latency-based strategy always assigns Apps' requests to their closest cloudlets without considering the workload in each cloudlet; DBC assigns Apps to the closest cloudlets until the workload of each cloudlet exceeds the average workload among cloudlets, without considering the diversity of applications in each cloudlet. Thus, both DBC and the latency-based strategy lead to a lower network delay and a higher computing delay than AREA.

AREA considers both the network delay of each App and the different types of workloads for each cloudlet in the workload allocation. To reduce the computing delay of all Apps, it tends to assign Apps with small computing sizes to the lightly loaded cloudlets. Furthermore, it also optimally allocates computing resources for different types of VMs based on their corresponding workloads, and thus significantly reduces the average response time per App. Meanwhile, as shown in Fig. 3, the average response time for different types of applications in AREA is significantly smaller than those of DBC and the latency-based strategy.

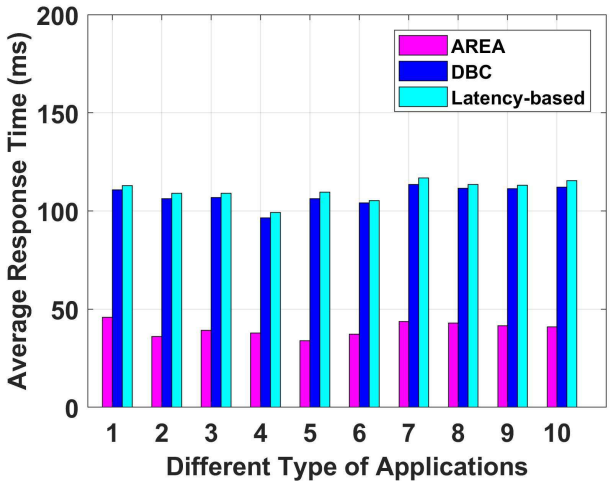

Fig. 3. Average response time for different types of IoT applications ($\lambda_{max} = 1.5$, $C_i = 2 * 10^8$).

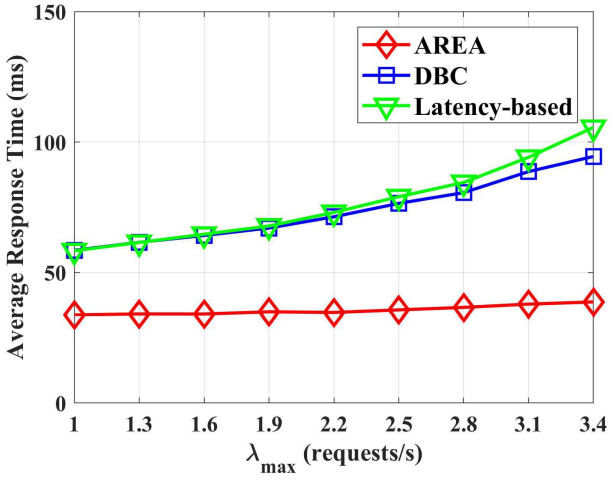

Fig. 4. Average response time with respect to λ_{max} ($C_i = 3.8 * 10^8$).

We further analyze how the workloads of Apps affect the performance of the three algorithms. Note that the value of λ_{max} reflects the workloads of Apps, i.e., increasing λ_{max} increases workloads of Apps. As shown in Fig. 4, with the increase of λ_{max}, the average response time of the three algorithms increases gradually. However, the average response time of AREA is much lower and increases more slowly as

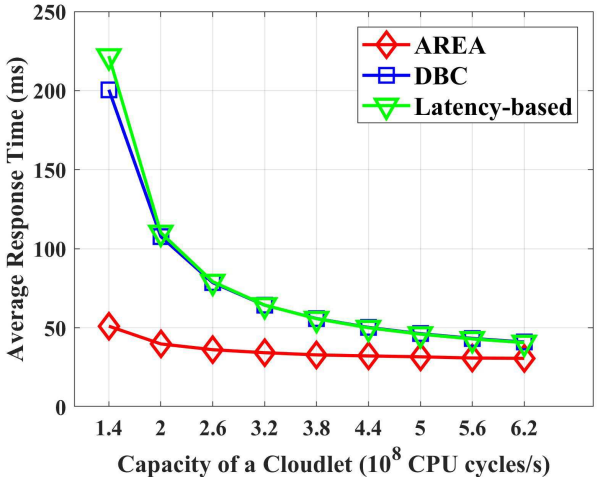

Fig. 5. Average response time with respect to the capacity of each cloudlet ($\lambda_{max} = 1.5$).

compared to those of the other two algorithms. When the workloads of Apps are heavy, AREA can always offload the App with the highest response time to an alternative cloudlet, and thus iteratively minimize the maximum response time among Apps. Meanwhile, AREA also optimally allocates the computing resources of each cloudlet to different types of applications based on their workloads and their corresponding computing sizes, and thus further reduces the computing delay.

Moreover, we investigate the impact of cloudlets' capacities on the average response time. Fig. 5 shows that the response time of the three algorithms when the capacities of cloudlets increase. It can be seen that AREA achieves much lower average response time when the capacities of cloudlets change. When the capacities of cloudlets are small, since DBC and the latency-based algorithm do not balance the workloads among cloudlets based on different types of applications (i.e., considering all task requests are homogeneous), AREA leads to a remarkably lower computing delay, and thus incurs lower response time. However, when the capacities of cloudlets are very high, the computing delay is no longer a dominating factor for the average response time, and thus the average response time of DBS and the latency-based algorithm get close to that of AREA.

We also analyze the impact of the number of UEs on the average response time of Apps. As shown in Fig. 6, the average response time of AREA increases much slower than those of the other two algorithms. Since AREA considers the difference between applications, it tends to assign Apps with smaller task sizes to lightly loaded cloudlets and allocates more computing resources to them, thus minimizing the average response time of all UEs' Apps. Therefore, as the number of UEs increases where the computing delay is the dominating factor in the average response time, AREA is able to achieve a lower computing delay than the other two algorithms, thus improving the performance of the average response time.

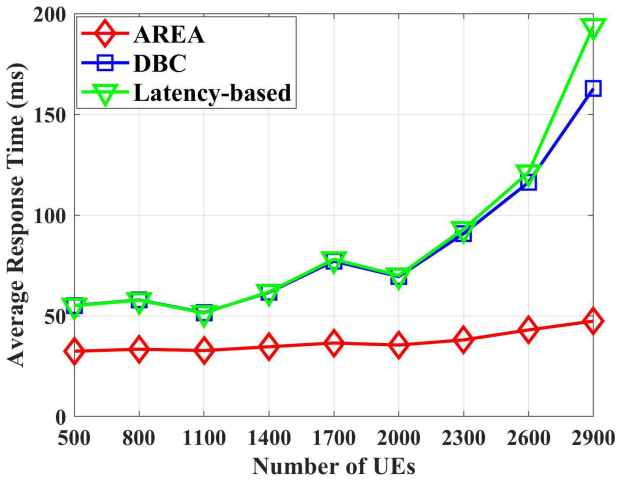

Fig. 6. Average response time with respect to different number of UEs ($\lambda_{max} = 1.5$, $C_i = 3.8 * 10^8$).

VII. Conclusion

In this paper, we have proposed the Application awaRE workload Allocation (AREA) scheme for edge computing based IoT. AREA assigns different types of workloads in each UE to their corresponding VMs in each cloudlet and optimally allocates the computing resources of each cloudlet to its application based VMs. We have formulated the problem of minimizing the average response time of Apps and designed the AREA algorithm to achieve a suboptimal solution. Simulation results have verified the performance of AREA.

References

[1] X. Guo, L. Chu, and X. Sun, "Accurate localization of multiple sources using semidefinite programming based on incomplete range matrix," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5319–5324, July 2016.

[2] L. Wang and R. Ranjan, "Processing distributed internet of things data in clouds," *IEEE Cloud Computing*, vol. 2, no. 1, pp. 76–80, 2015.

[3] N. Ansari and X. Sun, "Mobile edge computing empowers internet of things," *IEICE Trans. on Comm.*, vol. E101-B, no. 3, pp. 604–619, 2018.

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

[5] Y. Wang *et al.*, "Improving utilization through dynamic VM resource allocation in hybrid cloud environment," in *20th IEEE Intl. Conf. Parallel & Distributed Systems (ICPADS 2014)*, 2014, pp. 241–248.

[6] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *35th Annual IEEE Intl. Conf. on Comp. Comm. (INFOCOM 2016)*, San Francisco, CA, April 2016, pp. 1–9.

[7] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, 2017.

[8] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–6.

[9] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, 2016.

[10] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Inet. of Things J.*, vol. 3, no. 6, pp. 854–864, 2016.

[11] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the internet of things," *IEEE Comm. Mag.*, vol. 54, no. 12, pp. 22–29, 2016.

[12] M. Jutila, "An adaptive edge router enabling internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1061–1069, 2016.

[13] R. Deng *et al.*, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec 2016.

[14] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things," in *2017 IEEE Intl. Conf. on Edge Computing (EDGE)*, Honolulu, HI, June 2017, pp. 17–24.

[15] H. Zhang *et al.*, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Inet. of Things J.*, vol. 4, no. 5, pp. 1204–1215, Oct 2017.

[16] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Communications Letters*, 2018, DOI:10.1109/LCOMM.2018.2801866, early access.

[17] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. on Cloud Computing*, vol. 5, no. 4, pp. 725–737, Oct 2017.

[18] L. Yang *et al.*, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Trans. on Computers*, vol. 65, no. 5, pp. 1440–1452, 2016.

[19] Q. Fan and N. Ansari, "Green energy aware user association in heterogeneous networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'2016)*, Doha, Qatar, Apr. 2016.

[20] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *2014 IEEE Network Ops. and Mgmt. Sym.*, Krakow, Poland, May 2014.

[21] C. Yu *et al.*, "Software-defined latency monitoring in data center networks," in *Intl. Conf. on Passive & Active Net. Measurement*, vol. 8995, Mar. 2015, pp. 360–372.

[22] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[23] R. Landa *et al.*, "The large-scale geography of internet round trip times," in *IFIP Networking Conference*, Brooklyn, NY, May 2013, pp. 1–9.

Qiang Fan (S'15) received his M.S. degree in Electrical Engineering from Yunnan University of Nationalities, China, in 2013. He is currently a research assistant and a Ph.D. candidate in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, New Jersey, USA. His research interests include mobile cellular networks, mobile cloud computing, Internet of things, and free space optical communications.

Nirwan Ansari (S'78-M'83-SM'94-F'09) received his B.S.E.E. degree from NJIT, M.S.E.E. degree from the University of Michigan, Ann Arbor, and Ph.D. degree from Purdue University, West Lafayette. He is Distinguished Professor of Electrical and Computer Engineering at NJIT. He has also been a visiting (chair) professor at several universities. He recently authored Green Mobile Networks: A Networking Perspective (Wiley-IEEE, 2017) with T. Han, and co-authored two other books. He has also (co-)authored more than 500 technical publications, over 200 in widely cited journals/magazines. He has guest-edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, and various aspects of broadband networks. Some of his recognitions include several Excellence in Teaching Awards, a few best paper awards, the NCE Excellence in Research Award, the ComSoc AHSN TC Technical Recognition Award, the IEEE TCGCC Distinguished Technical Achievement 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 36 U.S. patents.